A Trust-Aware, Self-Organizing System for large-scale federations of Utility Computing infrastructures

F. Messina[a], G. Pappalardo[a], D. Rosaci[b], C. Santoro[a], G.M.L. Sarné[c]

[a]*Department of Computer Science and Mathematics, University of Catania, Viale A.Doria 6, 95126 Catania CT*
[b]*Department DIIES, University "Mediterranea" of Reggio Calabria, Loc. Feo di Vito, 89122 Reggio Cal. RC*
[c]*Department DICEAM, University "Mediterranea" of Reggio Calabria, Loc. Feo di Vito, 89122 Reggio Cal. RC*

Abstract

On-demand computing environments, like Cloud/Grid systems, consist of nodes that individually manage local resources intended to be served to clients. When a client needs some resources, it has the problem of finding the most suitable nodes capable of providing them. In addition, a provider node too may be in need to efficiently locate resources for itself, given the emerging, highly *competitive*, context of large-scale federations. Indeed, a node competes, with the other federated ones, to obtain the assignment of available tasks. To this end, it may decide to publish a set of resources/services wider than the one it has currently available. Should such a node be assigned a job for which its actual resources are insufficient, it could end up requiring the collaboration of other nodes.

Hence the crucial problem, for nodes and clients alike, to determine the most promising collaborators. For this purpose, in the competitive and demanding scenarioes considered, we advocate taking into account the *trustworthiness* of nodes in declaring their capabilities. I.e., to help it making an effective selection of possible collaborators, each node should be provided with a trust model for accurately evaluating the trustworthiness of its interlocutors.

In this paper, a trust-based approach for large-scale federations Utility Computing infrastructures is proposed. The proposed model is designed to allow any node to find the most suitable collaborators in an efficient way, avoiding exploration of the whole node space. A fully decentralized approach is employed, which allows nodes of a federation to be organized in an overlay network on the basis of suitable criteria. This enables any customer or provider in need of collaborators to determine a suitable set of candidate nodes within which to search.

Keywords:

Distributed Systems, Application and Expert Systems, Information Systems Application

Email addresses: {messina}@dmi.unict.it (F. Messina), {pappalardo}@dmi.unict.it (G. Pappalardo), Rosaci@unirc.it (D. Rosaci), {santoro}@dmi.unict.it (C. Santoro), sarne@unirc.it (G.M.L. Sarné)

1. Introduction

Since the development of computational Grids [18], started in the middle of $1990s$, virtualization technology [6, 34] and commodity hardware led to the development of the Cloud computational paradigm [19, 25], which is a practical example of a new distributed computing paradigm, i.e. *On-demand computing* [25]. The next step is represented by the construction of federated computing infrastructures, i.e. Grid Federations [48] and federated Cloud Computing infrastructures [9, 11, 38, 61]. For instance, we can cite the Reservoir model developed by IBM [51], in order to "*facilitate an open, service-based online economy in which resources and services are transparently provisioned and managed across clouds on an on-demand basis at competitive costs with high-quality service*", bringing a vision of "*implementing an architecture that would enable providers of cloud infrastructure to dynamically partner with each other [...] while fully preserving their individual autonomy in making technological and business management decisions*".

The increasing interest for large-scale federations of utility computing infrastructures is motivated by the opportunity for providers to mutually "rent" virtualized services and, in turn, allocate unused resources in order to increase profits [22]. That is one of the main reasons to have interest in federating resources: establishing a *collaborative* context at large-scale will help to achieve the above objectives. For instance, providers of computing infrastructures (IaaS, PaaS, STaaS) are able to provide services like *auto-scaling, advanced monitoring, automation in service deployment* to providers which make their business by offering high level e-services (e.g. SaaS, DaaS) [5]. In such a collaborative scenario, *fulfilling computational requirements* of user requests *as efficiently as possible* becomes crucial: this is a typical problem in a large-scale distributed system, in which resources are shared by several different organizations [37]. For instance, a recent study [62] deals with the problem of balancing profit, price and QoS in a typical federated environment, by proposing policies to help making decisions aimed at outsourcing services, contributing to the federation or even terminating spot VMs to reject less profitable requests.

Another important issue to address is that service providers – while having interest in joining a federated network that generally results more attractive for users than a single provider – are in *competition* with each other, in order to offer a number of *interchangeable* on-demand services at any level. The increasing trend of offering interchangeable services by federated providers leads to the interesting recent development [32] of Cloud Service Composition, a popular choice affording clients the opportunity of choosing among several alternatives. Yet, in a highly competitive environment, this opportunity carries a risk, in that nodes *can declare to provide a better QoS than currently possible*, as providers have a "natural" tendency to gamble with their own *reputation* [22]. Since Cloud customers tend to prefer paying services in the face of signed SLAs (Service Level Agreements) [3, 67], when negotiated QoS is not fulfilled, both the node provider and the user are penalized. While the former will generally pay a refund [3], the activities of the latter (the customer) will be affected by a measured level of QoS which is not compliant with the SLA, which may lead to a loss of revenue [5] uncompensated by the refund. In order to avoid penalties and losing reputation with clients, a node, unable to fulfill an assigned client request, may seek collaboration from

other nodes at a price, effectively outsourcing, in turn, some of the services it should provide. However, in so doing, it may incur the hazard of not finding collaborators that are able to provide a sufficient level of reliability themselves. In this scenario, an accurate selection of interlocutors, aimed at *finding the most promising collaborators*, can make a great difference. As we discuss below, this is a major issue we deal with in this work, i.e. to provide a decentralized support to assess the trustworthiness of Cloud providers in a federated context.

Given these premises, we propose and discuss a decentralized approach, called *Hypertrust*, for federated, large scale utility computing infrastructures, aimed at providing a twofold support to customers and providers. First of all, Hypertrust deals with trustworthiness in a competitive, large-scale distributed system. The basic idea is to *design a trust system* suitable to support the nodes in choosing their interlocutors, *limiting the search space* to only those nodes declaring to have the suitable resources for the collaboration. The second feature is motivated by the fact that, in spite of existing trust models like RRAF [20, 53] were introduced to combine several different concerns related to trustworthiness of computational nodes into a unique synthetic trust measure, they demonstrate good performance when applied to a small/medium-size system and, when the system includes a very large number of nodes, as in the case of large-scale federated computing systems, this selection task becomes infeasible.

The last target, i.e. *fulfilling computational requirements of user requests as efficiently as possible* is pursued through a *decentralized* technique, intended to organize computing nodes as peers of an overlay network whose "small world-like characteristics make the resource finding process both effective and efficient. Indeed, the overlay construction algorithm is guided by *resource status similarity*, so that peers featuring a similar amount of resource availability tend to be interconnected (in terms of overlay links), thus forming clusters which, in turn, are connected together though sparse "long links[1]. Not surprisingly, resource finding by navigating such an overlay network, turns out to be rather efficient. This is how we address the first issue of (*efficient resource finding*).

Some special nodes are also introduced, playing the role of *Task Allocators* (*TA*): they are mediators, provided in principle by each organization, which collect users' feedbacks and, based on these, select a suitable node for a service. As TAs collect feedbacks about service requests for which they have been contacted, it makes sense for users to generally rely on them, thus exploiting previous user experiences. In contrast, a computational node, in need to select backup collaborators to provide a service, will resort to our decentralized resource finding algorithm to determine a set of candidate nodes, and then exploit the trust model, to choose a collaborator.

We simulated some experimental scenarioes in order to highlight the advantages introduced by our proposed approach. Results prove that, in a competitive scenario, recourse to the Hypertrust overlay network, enhanced with the trust model for interlocutor selection, affords a significant advantage with respect

[1] Such an organization resembles the classical model of small-world networks [66].

to selection based on resource declaration alone. Moreover, as discussed in Section 5.6, not only do TA components bridge the gap between the two models performing the selection needed by the Hypertrust selection system: they also bring a further contribution, in terms of QoS, to the final user. We have also performed an experimental analysis of our decentralized resource finding algorithm, employed to seek collaborators with suitable characteristics. Aptly, it has shown quite a high efficiency, despite the lack of any centralized directory or index.

The remaining of the paper is organized as follows. Section 2 introduces the competitive, large-scale distributed scenario we deal with. Section 3 describes the adopted trust model and brings out a brief discussion about existing models. In Section 4, the Hypertrust overlay construction and resource finding algorithm is discussed, with particular emphasis on Cloud Federations. In Section 5 we provide an experimental evaluation of the proposed approach. Related work is presented in Section 6. Finally, Section 7 reports our conclusions and discusses possible developments of our ongoing research.

2. The Competitive Hypertrust Scenario

An increasing attention is emerging about the dynamics of resource pricing in multi-cloud/grid contexts where, similarly to other competitive utility markets, each provider node is driven by profits, while their clients seek lower costs and/or higher quality-of-service.

Several works deal with this topic, but some of them completely ignore the competitive nature of distributed computing environments, like [64], which discusses a technique to find a market equilibrium between two different set of providers, or [10] and [69], where authors respectively optimize service queues and profits, or [46], where a hierarchical cooperative game models cooperative providers. More realistically, most providers act in a competitive way, to maximize profits in a context where each node is a direct competitor of the other nodes, in that it provides similar services. Often such scenarios are modeled by a Nash equilibrium [45], in which each provider gives its best response to the strategies of its opponents, as in [65] where a normal-form game is proposed to prove the existence and uniqueness of the Nash equilibrium to provision and manage SLA-based resources among different providers. However, other approaches not explicitly involving game theory are available in the literature as, for instance, [52], which is based on economic models of demand-offer equilibrium, or [60], which tries to maximize profits based on right pricing and the provider's data-center size.

The novelty and originality of our proposal lie in our considering a *competitive* distributed scenario, where nodes belonging to different domains and geographical locations: i) advertise *pay-per-use* services in a competitive fashion, and ii) *cooperate* in order to satisfy a node service request. Therefore, in this competitive/cooperative context, it is assumed that: (*i*) each service is associated with a category c stored in a catalog $\mathcal{C}$; (*ii*) the availability of software and computational resources (e.g. CPU time, RAM, disk space) changes over time. Resource availability is managed and published by local resource managers, as PaaS/IaaS middlewares [72] coupled with suitable VMMs [6, 34] or conventional job schedulers which,

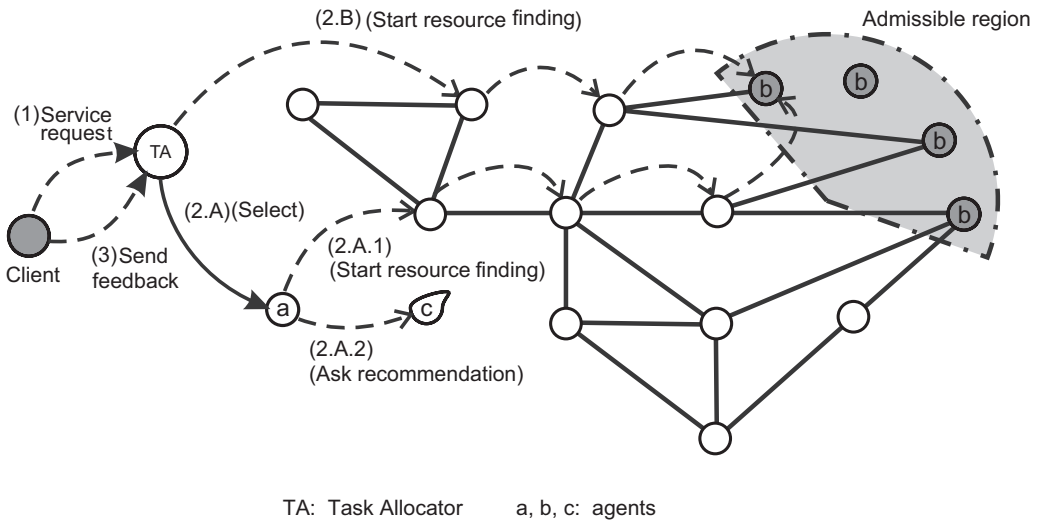


Figure 1: The Hypertrust scenario: (A) - The address of a Task Allocator (TA) is known to the client and the request is sent to it; (B) - The address of the TA is not known or the TA is not alive. The client will gossip the request to its own neighbors to find a TA.

by sharing computational resources within Grid Virtual Organizations [18], are also able to manage and publish the catalog $\mathcal{C}$ in a distributed way. Nodes are organized in an *overlay network* featuring certain characteristics and constructed by using the algorithm explained in Section 4; therefore, each node is interconnected with one or more *neighbors*.

2.1. Interaction Model

We assume that, for each administrative domain, there is a special node which offers a "mediator service" to the other nodes of its domain; we call it the *Task Allocator* (TA): it is the main interface for clients and performs the handling of their requests. We assume that, in order to improve the level of fault tolerance, the role of TA is replicated on several nodes of the same domain, but, at a given time, only one TA can be active for each domain. Whenever a client needs a certain service, it starts an interaction protocol which involves all the entities of our scenario; the interaction dynamics is depicted in Figure 1 and detailed in the description below.

Let us to consider a client which asks for a service s^c associated with a category $c \in \mathcal{C}$. The first operation performed by the client is the selection of an organization which could provide the service; subsequently, the client contacts the TA of that organization and sends the request to it. This initial interaction is represented by step (1) in Figure 1. We suppose that the TA is known to the client; if this is not the case, it can be searched for using a *gossiping algorithm* detailed in Section 2.2.

The TA, after receiving the request, performs a selection of a possible node that, in the opinion of TA, can be the most suitable for executing the task (step (2.A) of Figure 1); this selection is made by using the knowledge, gathered by the TA itself, about the effectiveness shown in the past by the various nodes, in providing the services of category c. The basis of such a selection is a *preference index* which is a result

of the feedbacks gathered by the TA about the services offered in the past. Such information are properly handled by the trust and reputation model which is discussed in Section 3. In particular, the selection is based on the P_{TA} index computed according to Section 3.5.

The selected node, say a, will receive the request forwarded by the TA (step (2.A) of Figure 1) and, in turn, may decide:

1. To directly serve the request, given that it has enough resources; in this case, the request is assigned to a which will execute the task according to the specifications.
2. To ask the collaboration of another node, if it *does not* have the adequate amount of resources but wants, in any case, to provide s^c.

In the second case, node a will need to contact other nodes which could provide the missing amount of resources; to this aim, it starts a *resource finding process* which is represented by the step (2.A.1) of Figure 1. The algorithm exploited in this case, which is detailed in Section 4, has the objective of finding the *set of possible nodes* which possess (or claim to possess) the required amount of missing resource for s^c. Such a set is referred as the *admissible region*; the reason why it is named as a "region" (a term which suggests some spatial considerations) will be made more clear in Section 4, where the overall resource finding protocol will be described in-depth.

Since we are in a *competitive environment*, during this search process, contacted nodes could announce the availability of the requested resources without really having all or part of them. If a request is assigned to a node falling into this case, a possible outcome could be a service provisioning having a quality lower than that promised. To avoid it, once the admissible region is found and a node b has been selected as the best candidate, then node a, in order to confirm this choice, can ask to a third node, say k, a *recommendation* about the expertise of b (step (2.A.2) of Figure 1). As explained in Section 3, this *recommendation* represents a subjective index of the *Quality of Service* (QoS) provided by b in the past, as it is perceived by k. After the service has been supplied by the selected nodes (e.g. nodes a and b), the TA receives a *feedback* from the client, that is an evaluation of the QoS of any node involved into the service provided to the client. This phase is referred as step (3) of Figure 1. This way, node a, which provided the service, updates its internal state (*trust model*) about the set of collaborators (i.e. node b), and the TA updates its own trust model, with respect to c, which is used to evaluate the *preference index* for further requests on services belonging to c.

2.2. Task Allocator Dynamics

The TA plays a central role in the proposed scenario because it holds the (updated) information needed to perform a direct and aware selection of the node a; moreover, for the client, the TA represents the reference node to which sending the requests. We therefore assume that: *(i)* the TA must possess the adequate amount of information to allow a proper selection of node a, and *(ii)* a live TA must be always found by any client.

Assumption *(i)* is not true when a TA is started for the first time and also when it consider the information of its own trust model not enough (or too bad) to make an adequate selection. In both of these situations, the TA cannot perform a direct selection but it has to find a suitable node; to this aim, it starts a resource finding process (step (2.B) of Figure 1), as specified in Section 4.2. Also in this case the selected node can look for one or more collaborators within the admissible region and can ask recommendation as discussed above (step 2.A.2 of Figure 1).

The second assumption implies that clients should always know a TA. If this is not the case, a simple gossip-based protocol, inspired from [31], is exploited. To this aim, the client prepares a *TA search message*, which includes its *ClientAddress*, a unique *MessageID* and a *Time-To-Live*, and sends it to one or more of the nodes it knows[2]. If the node receiving the message is a *TA*, it answers directly to the client. Otherwise, it performs the following operations:

1. If a message with the same *MessageID* has been already received, it is simply discarded.
2. The *Time-To-Live* field is decremented; if it has reached zero, the message is discarded.
3. Otherwise, the node forwards the message to its neighbors with a probability v which is called *gossiping threshold*.

In this way, as we further discuss in Section 5.3 an evaluation of the algorithm reported below, a peer able to contact a Task Allocator can be found in a very few time-steps even when a high number of TA leave from the network. The case described above[3] is depicted as *(B)* in Figure 2.

TA Finding Gossip Protocol.

```
1:  if gossip message arrived (ClientAddress, MessageID, TTL) then
2:    if node is the TA then
3:      send OK to ClientAddress
4:    end if
5:    if the MessageID has been already received then
6:      Discard message
7:    else
8:      TTL ← TTL − 1
9:      if TTL = 0 then
10:       Discard message
11:     end if
12:     for node ∈ neighbours do
13:       if random_uniform(0, 1) < v then
14:         send (ClientAddress, MessageID, TTL) to node
15:       end if
16:     end for
17:   end if
18: end if
```

3. The Trust Model

[2] We suppose that the client knows *at least one* of the nodes of the system, no matter its role.

[3] We also report, in Figure 2, for convenience, step (1) of Figure 1 as step (A).

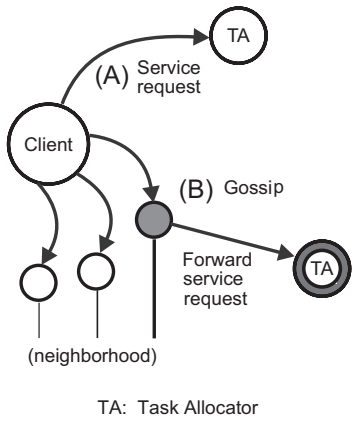


Figure 2: Contacting the Task Allocator TA for a service request: (A) - The address of a Task Allocator (TA) is known to the client and the request is sent to it; (B) - The address of the TA is not known or the TA is not alive and, therefore, the client exploits a simple gossip-based protocol for forwarding its service request to a TA.

In the context of e-services, when two agents interact with each other, one of them (the truster) assumes the role of a service requester and the other (the trustee) acts as an e-service provider. According with [21, 24], this study considers trust as the set of quantified beliefs by a truster with respect to the integrity, benevolence, competence and predictability of a trustee in a specific context.

In this scenario, the trust given by the truster to the trustee can be based on the combination of *reliability* and *reputation* measures. The former is a direct measure derived by subjective observations (or perceptions) obtained by the truster during its direct interactions with the trustee; the latter is an indirect measure based on the opinions (or recommendations) about the trustee coming from the other actors of the community which act as witnesses or third parties with respect to both the truster and the trustee.

In the literature, several trust models have been proposed for representing both reliability and reputation [24, 56, 57]. The RRAF model [20, 53], for instance, was introduced to suitably combine these two measures into a unique synthetic trust measure. In RRAF, each node selects, among all the nodes, the most promising collaborators based on such a trust measure. However, although RRAF demonstrated a good effectiveness when applied to a small/medium-size system, when the size of the system becomes too large with a very large number of nodes, as in the case of large-scale federated computing systems, this selection task becomes infeasible.

In this Section we present the trust model of Hypertrust; it is based on the combined computation of reliability and reputation, and an additional preference (i.e. trust index) is introduced for the Task Allocator, as discussed in Section 3.5. To model the trust system, we denote $\mathcal{A}$ as the set of all the nodes belonging to the Hypertrust network, and let a_i be the i-th element of $\mathcal{A}$. Each node a_i maintains a set of five mappings, each of them related to measure a specific aspect of the trust model; each measure is represented by a number ranging in $[0, 1] \in \mathbb{R}$, where $0/1$ represents the minimum/maximum value. Measures belong to two different categories, i.e. **Reliability/Recommendation** measures and **Preference** measures. The

former set includes the following mappings:

$SR_i(j, c)$: it is the *Service Reliability* that a_i assigns to a node a_j for services belonging to category $c \in C$ by taking account its past direct experiences occurred with a_j. A value $0/1$ means that a_j is evaluated as completely unreliable/reliable.

$RR_i(j, c)$: it is the *Recommendation Reliability* that a_i assigns to the *opinions* provided by a_j that are referred to services belonging to $c \in C$. It measures the reliability for a_i of the gossip coming from a_j about the other nodes belonging to $\mathcal{A}$ with respect to a category of services c; in other words $RR_i(j, c)$ represents how a_i weights the recommendations provided by a_j.

$R_i(j, c)$: it is the *Reputation* assigned by a_i to a_j, based on some recommendations coming from other nodes of the community and represents how the community perceives the reputation of a_j with respect to services associated with c. A value $0/1$ means that a_j is evaluated by the other nodes of the community as completely unreliable/reliable.

The latter set, i.e. **Preference** measures, includes the following mappings:

$\gamma_i(j, c)$: it is the *reliability preference*; it represents the *weight* that a_i assigns to the value of service reliability (SR_i) with respect to the reputation (R_i) in evaluating the overall trust score referred to a_j and the service $c \in C$. As we discuss later, the percentage of relevance to give to the reliability is represented by the value $\gamma_i(j, c)$, while the percentage to assign to the reputation is given by $(1 - \gamma_i(j, c))$. The value of $\gamma_i(j, c) \in [0, 1]$ can be fixed by each node.

$P_i(j, c)$: it is the *overall trust score* that the *node* a_i, based on its perceived reliability and reputation, assigns to a_j.

The TA, together with the mappings described so far, manages also another measure, called $P_{TA}(j, c)$, and represents the overall trust score that the TA itself, based on the collected feedbacks, assigns to a node a_j, concerning services belonging to category c.

3.1. Computation of SR mapping

As discussed in Section 2, in the step (3) of Figure 1, the TA collects the feedbacks for the service s provided by a node a_i; since a_i might have asked a collaboration to other nodes, such feedbacks represent the quality of not only a_i but also of the collaborator nodes a_j. In turn, such feedbacks are forwarded by the TA to the node a_i which updates its own mapping $FEED_i(s, j) \in [0, 1] \in \mathbb{R}$. A feedback equal to $0/1$ means the minimum/maximum quality of the service.

Let $Services_i(j, c)$ be the set of services provided by a_i with the collaboration of a_j at the previous step, then the *current service reliability* $sr_i(j, c)$ shown by a_j is computed as:

$$sr_i(j,c) = \frac{\sum_{s \in Services_i(j,c)} FEED_i(s,j)}{||Services_i(j,c)||}$$

where $sr_i(j,c)$ is the service reliability of node a_j computed by the node a_i with respect to all the services s assigned to the node a_i for which the node a_j provided a collaboration, i.e. $Services_i(j,c)$.

For instance, let us to consider a situation in which the node a_i receives three services in the category c from the node a_j, and let $FEED_i(1,j)$=1, $FEED_i(2,j)$=0.2, $FEED_i(1,j)$=0.3 be the three corresponding feedbacks. In such a situation, we compute $sr_i(j,c) = (1 + 0.2 + 0.3)/2 = 0.5$.

The next step performed by a_i is the updating of the *Service Reliability* SR_i, by averaging its value at the previous step and the *current service reliability* previously computed:

$$SR_i^{(t)}(j,c) = \alpha \cdot SR_i^{(t-1)}(j,c) + (1-\alpha) \cdot sr_i(j,c)$$

where $\alpha \in [0,1] \in \mathbb{R}$ represents the relevance that a_i gives to the past evaluations with respect to the current contribution. In other words, α measures the relevance given to the *memory* with respect to the current time. Note that in the Hypertrust model α does not change by considering the different categories of $\mathcal{C}$. For instance, suppose that at the time $t-1$ the service reliability $SR_i^{(t-1)}(j,c)$ was equal to 0.8, and that the currently service reliability is that previously computed in the example above, i.e. $sr_i(j,c) = 0.5$. Also suppose that $\alpha = 0.5$, giving the same importance to both past evaluations and current contribution. In this situation, the new service reliability $SR_i^{(t)}(j,c) = 0.5 \cdot 0.8 + 0.5 \cdot 0.5 = 0.65$

3.2. Computation of RR.

$RR_i(j,c)$ represents the *reliability* assigned by a_i to the *recommendations* provided by a_j. This value is computed by properly comparing the recommendations received by a_i and the feedbacks, about the same nodes, collected by the client/TA. To this aim, a further mapping for collecting recommendations, $RECC_i(k,j,c)$, is defined; it contains the recommendations received by a node a_i during step (2.A.2) of Figure 1. Also in this case, each value is in the range $[0,1] \in \mathbb{R}$ and represents the recommendation that a_k provided to a_i about a_j with respect of a service $c \in C$.

On this basis, the *Recommendation Reliability* is updated as follows:

$$RR_i^{(t)}(k,c) = \beta \cdot RR_i^{(t-1)}(k,c) + (1-\beta) \cdot (1 - err_i(k,c))$$

where

$$err_i(k,c) = \sum_{\substack{j \in S_i(k,c) \\ s \in Services_i(j,c)}} \frac{|RECC_i(k,j,c) - FEED_i(s,j)|}{\|S_i(k,c)\| \cdot \|Services_i(j,c)\|}.$$

Here, $S_i(k,c)$ is the set of the nodes of the community that provided a collaboration to a_i for services belonging to category $c \in C$, and such that node a_k has provided a recommendation to a_i; $Services_i(j,c)$ is the set of services provided by a_i with the collaboration of a_j at the previous step; $\beta \in [0,1] \in \mathbb{R}$, has the same meaning than for the coefficient α.

For instance, suppose that a_i received recommendations from a_k about two nodes, namely $j1$ and $j2$, and that the recommendation about $j1$ is $RECC_i(k,j1,c) = 0.4$ while that about $j2$ is $RECC_i(k,j2,c) = 0.8$. Also suppose that the feedback for the actual services provided by $j1$ is $FEED_i(s,j1) = 0.2$ and that for the services provided by $j2$ is $FEED_i(s,j2) = 0.3$. In this situation, $err_i(k,c) = (0.2+0.5)/2 = 0.35$. If the recommendation reliability $RR_i^{(t-1)}(k,c)$ at the time $t-1$ was equal, for instance, to 0.7, and the coefficient β is equal to 0.5, the new recommendation reliability a the time t will be $RR_i^{(t)}(k,c) = 0.5 \cdot 0.7 + 0.5 \cdot 0.35 = 0.525$

3.3. Computation of R.

The *Reputation* $R_i(j,c)$ that a_i has of another node a_j, with respect to c, is then computed as the average of all the recommendations received by the other nodes of the community, suitably weighted by their *Recommendation Reliability* in order to hinder malicious behaviors:

$$R_i(j,c) = \frac{\sum_{k \in AS_j} RR_i(k,c) \cdot RECC_i(k,j,c)}{\sum_{k \in AS_j} RR_i(k,c)}$$

where $AS_i(j,c)$ is the set of the nodes of the community that provided a_i with a recommendation about a_j for services falling in $c \in C$.

For instance, suppose that a_i received recommendations about the node a_j with respect to c from two nodes, namely $k1$ and $k2$, and that these recommendations were $RECC_i(k1,j,c) = 0.4$ and $RECC_i(k2,j,c) = 0.8$. Moreover, suppose that the recommendation reliability of $k1$ is $RR_i(k1,c) = 0.2$ while that of $k2$ is $RR_i(k,c) = 1$. In such a situation, the reputation that a_i assigns to a_j with respect to c is $R_i(j,c) = (0.4 \cdot 0.2 + 0.8 \cdot 1)/(1.2) = 0.73$

3.4. Computation of P

As discussed in Section 2, each node a_i, at the end of the resource finding process —see step (2.A.1) of Figure 1— has to select a set of nodes within the admissible region which are the supposed most suitable candidates for requesting a collaboration. To this aim, a_i has to exploit the mapping P_i which computes the overall preference in the node a_j by taking into account both the service reliability $SR_i(j,c)$ and the reputation $R_i(j,c)$ as:

$$P_i(j,c) = \gamma_i(j,c) \cdot SR_i(j) + (1 - \gamma_i(j,c)) \cdot R_i(j,c)$$

where the value of the mapping $\gamma_i(j,c) \in [0,1] \in \mathbb{R}$, is used to weight the service reliability SR with respect to the reputation R.

For instance, if a_i computed the reliability and reputation of a_j as described in the example above, i.e. $SR_i(j) = 0.65$ and $R_i(j,c) = 0.73$, and we suppose that $\gamma_i(j,c) = 0.2$ (giving more importance to the reputation with respect to the reliability), then we have $P_i(j,c) = 0.2 \cdot 0.65 + 0.8 \cdot 0.73 = 0.71$

3.5. Computation of P_{TA}.

As explained in Section 2 the TA is a mediator which assigns client requests, receives feedbacks from clients and forwards them to the nodes involved in the service provisioning.

In particular, in order to select (*i*) a node —step (2.A.2) of Figure 1— or (*ii*) a set of nodes —step (2.B) of Figure 1— the TA computes its own preference $P_{TA}(j,c)$ with respect to the category c by considering all the feedbacks received by the clients in the past for the node j and referred to services belonging to c.

Formally, let $s \in c \in C$ be the generic service assigned to node i, and $\{j_1, j_2, \ldots, j_n\}$ be the set of involved nodes with $n \geq 1$ and $(i == j_k)$, where $k \in (1,n)$. Note that i is included into the list of nodes which have provided a contribution for the service s for convenience. Since each node in the set $\{j_1, j_2, \ldots, j_n\}$ provides a part of the service s, at the end of the service provisioning the client sends the feedback mapping $\{FEED(s^c, j_l) \in (0,1),\ l = 1 \ldots n\}$ to the TA.

Given that any single node i_l does not give the same contribution for the service s, the overall feedback is calculated by the TA as:

$$ServFEED(i,s) = \sum_{l=1}^{n} w(s,j_l) \cdot FEED_i(s,j_l)$$

where $w(s,j_l) \in (0,1)$ is the contribution given by the node i_l for the service s, and $\sum_{l=1}^{n} w(s,j_l) = 1$.

At a given step, the overall feedback for the services falling in c for which the node j was responsible at the previous step, is obtained by averaging over the values of $ServFEED(s)$ with $s \in c$:

$$sr_{TA}(i) = \frac{1}{\|Services_i(c)\|} \sum_{s \in Services_i(c)} ServFEED(i,s)$$

where $Services_i(c)$ is the set of services assigned to node a_i in the last step. The overall trust score for the node i is calculated by the TA as:

$$P_{TA}^{(t)}(i,c) = \delta \cdot SR_{TA}^{(t-1)}(i,c) + (1-\delta) \cdot sr_{TA}(i,c)$$

where $\delta \in (0,1)$ has the same meaning of α and β that have been explained before.

Note that the trust evaluation methods we propose here are similar from the analogous techniques developed in recommender systems field. However, in this first version of our approach, in order to evaluate

a recommendation provided by a node, we only use a content-based approach, i.e. we compare the recommendation with the actual result associated with that recommendation, as observed a posteriori. We do not use more sophisticated techniques, as collaborative-filtering or item-based approaches. As for the future, we plan to evaluate the implementation of other recommendation techniques for improving the precision of our evaluation.

4. Hypertrust overlay network and Resource finding

In this section we discuss the resource finding procedure which is part of the proposed model. Its basis is the construction and the maintenance of an overlay network, which eventually expresses convenience properties allowing a fully decentralized resource finding approach.

The proposed procedure has been developed and studied by the authors in the recent years [42, 43], and eventually interleaved in the wider context of federated utility computing infrastructures. We discuss the construction of the Hypertrust overlay network – along with the correlation of resource attributes with issues related to the trustworthiness of the nodes – in Section 4.1, and the resource finding algorithm in Section 4.2.

4.1. Overlay Construction Algorithm

The topological model of the Hypertrust overlay network is based on a suitable $n-$dimensional space (or *hyperspace*) where each *coordinate* represents the available quantity *declared* by the nodes for a given resource or any attribute value for a specific service or resource. Figure 1 shows an example of such a network with two dimensions. Each node is represented as a *point* in the hyperspace, and its position will change due to any coordinate variation, i.e. allocation or release of resources, or any changes in the statement policy of the node itself. The "distance" between two nodes is computed as the Euclidean distance among node's coordinates, and represents a measure of how much two nodes are "far" in terms of resource availability.

Overlay construction is a key aspect of Hypertrust and it is performed by means of a decentralized algorithm which runs on each node of the network, by executing the following steps:

1. let n^* be a generic node; node n^* contacts its linked$\big/$neighbour nodes in order to obtain, in turn, their linked nodes; this operation allows a node to obtain the set of L linked nodes at *2-hops*;
2. the set L is ordered by using the Euclidean distance of each node from n^*;
3. on the basis of some threshold parameter k and h, the node n^* rearranges its links, interconnecting itself with at least k near nodes, not most than h nodes. Node n^* could exceed the threshold h

whenever the so-called "essentially critical neighbors"[4] as discussed in [42], have to be preserved.

The steps above are executed by each node *continuously* (with a given period) in order to let the links organize properly; moreover, since a node, during its life, may change the quantity of declared resources, its position in the hyperspace will change. The continuous execution of the overlay construction algorithm will re-arrange its link in order to preserve the overlay network characteristics.

As detailed and proved in [42], the effect of the algorithm described above is twofold. Since the nearest nodes are preferred for link organization, the network exploits the formation of *communities* or *clusters of nodes* featuring a short intra-cluster distance. Moreover, keeping *long links* between clusters allows the system to preserve its connectivity. Since the Euclidean distance is a measure of resource availability similarity, such clusters are characterized by nodes with a resource status very closed to each other.

By exploiting short links within clusters, a fast navigation it is possible, e.g. to refine a resource finding process, while by using long links it is possible to quickly reach the region (i.e. the cluster) in the hyperspace where the nodes offering the requested resources reside.

In other words, the resulting overlay network features a structure quite similar to a *small-world* network [66]; as it is known, such networks exhibit a high clustering degree and a very low average path length, characteristics which are very important to make resource finding effective (for the evaluation see Section 5.7).

Figure 3 shows some steps in the construction of a bi-dimensional simulated overlay network. In this case, we started the simulation from a connected random network with a set of nodes whose coordinates were uniformly distributed in the hyperspace. Nevertheless, it can be verified that by constructing an overlay network over an hyperspace with an initial "skewed" distribution (e.g. clusters having the same values/attributes for resources) eventually leads to a network with the characteristics discussed above.

4.2. Mapping federated infrastructures into Hypertrust networks

Tables 1 and 2 reports samples of attributes which can be used for building Hypertrust network. In particular, since attribute describing services and computational resources are different in nature, a common solution is represented by the construction of several Hypertrust networks enclosing nodes of Grid Virtual Organizations [18], federated Grid VOs [48], federated Cloud providers [11, 38], and so forth. Therefore, a Hypertrust overlay network can be constructed among a number of federated Data Centers by offering IaaS services or by mapping a number of relevant attributes of interoperable PaaS services made available by a number of federated Cloud providers. Moreover, due to the decentralized nature of the resource finding and overlay network (described in Section 4.1), the resulting overlay networks can be maintained by high level of efficiency, and several parallel finding process can be performed to find suitable nodes for the customer,

[4] This solution ensures that the network will never partition, a situation which is highly undesirable, since it would imply that a set of nodes became unreachable.

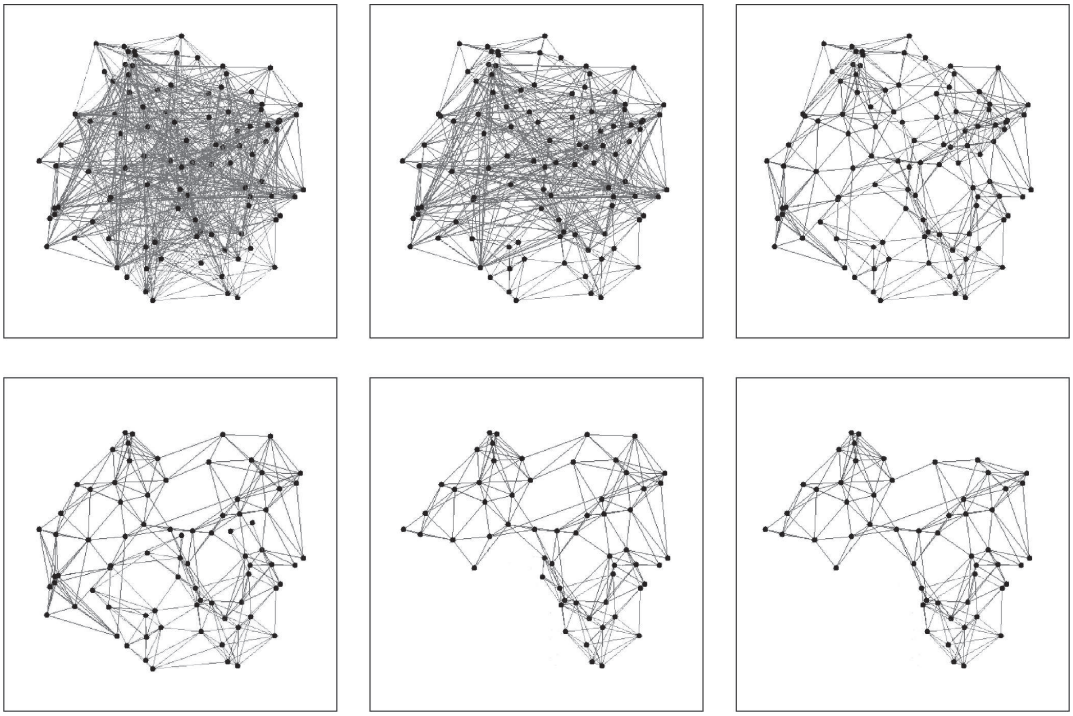


Figure 3: The overlay network construction

Infrastructure	Measure / attribute	Infrastructure	Measure / attribute
Grid VO	No. of available CPUs	Grid Federation	No. of available CPUs
	Available memory		Available memory
	Average waiting time (scheduler)		Level of interoperability
	Available disk space		VO scheduling policy

Table 1: Sample parameters for large scale Grid infrastructures

even the number of nodes is very high (see also Section 4.2).

4.3. Resource Finding

A so-formed Hypertrust network is the basis for the interaction steps (2.A.1) and (2.B) of Figure 1, in which a set of nodes able to fulfill a certain resource/job request needs to be found. Indeed, as a Hypertrust network can ensemble, in principle, heterogeneous nodes belonging to different infrastructures (cf. tables 1 and 2), a request may be a Grid job submission, a request for a set of VMs or a request for a composition of PaaS/SaaS services. Each request is modeled by means of a tuple containing a set of values for different resource attributes, each one appropriately encoded in a suitable numerical domain. As a consequence, a *resource request* can be described as a point in the hyperspace, which represents the lower-left corner of a *semi-space* or the *admissible region* whose internal nodes are those *claiming* some resources having suitable attributes. The decentralized approach for resource finding we present in this

Measure / attribute	**Service type**	**Example**
Availability	XaaS	99.5%
Reliability	XaaS	88.0 %
Response time	SaaS, PaaS	95th percentile $\leq$ 100ms
Network bandwidth	XaaS	100 MiB/s upstream, 1GiB downstream
Average processing capability	SaaS	$\geq$ 100 transactions per minute
Development & testing tools	PaaS	App Engine Environment
Open Platform	PaaS	Various REST-styles APIs
OS	PaaS	Linux
Development environment	PaaS	Specific Header/libraries
Software licenses	PaaS	Matlab for 100 CPUs
Max #CPUs per VM	IaaS	16
Max amount of memory per VM	IaaS	1 GiB
Virtualization platform	IaaS	Xen, VMware
Inter-Cloud migration Capability	IaaS	APIs, agreements with third party
Auto Scaling	IaaS, PaaS	Level of responsiveness and accuracy
Security measures	IaaS	VMI capabilities, secure code execution[13]

Table 2: Sample parameters for large scale Cloud Federations

Section is based on the following *check-and-forward* procedure:

1. a node receiving the request checks its capability to *fulfill* it; in this case the node belongs to the admissible region, so we reached the target and can continue with step 4;
2. if the node does not *declare* suitable resources, it contacts its neighbors and, on the basis of their resource status, forwards the request to one of them, selected by using an appropriate heuristic; such an heuristic is chosen in order to help the request to reach the admissible region as soon as possible;
3. the algorithm keeps track of all the nodes visited (this set is carried out together with the request), if all the possible nodes to jump onto are already analyzed, the system does not include a node suitable to host the request, so the algorithm terminates with failure;
4. when we found a node belonging to the admissible region, by suitably navigating through links, we can reach other valid nodes, in order to build the set needed by the reputation model.

The decentralized approach presented above can be formally described by the pseudo-code shown in Algorithm 1, by which we assume that a Hypertrust network built over the nodes of a Cloud Federation [38]. Algorithm 1 is composed of two distinct phases, called *SLA-Testing & Allocation* (SLA-TA)

and *Forwarding* (FWD), respectively identified by lines (1-23) and (24-33). In the SLA-TA phase, the algorithm checks whether the node, or one of its neighbors, exposes a set of characteristics matching with the requirements of the customer. Line 1 of Algorithm 1 assumes that the request has landed into the node n to start services/allocate resources for request $\overline{q}$. The SLA negotiation [3] is modeled by means of the function *testSLA*$(x, \overline{q})$, which is used to perform, on node x, a full check on the potential agreement on the QoS parameters for the services required to host the request $\overline{q}$.

Algorithm 1. Resource finding procedure at node n for request $\overline{q}$, allocation heuristic S_{alloc}, forward heuristic S_{fwd}, neighbors N.

1: $R_{\overline{q},n} \leftarrow testSLA(n, \overline{q})$
2: **if** $(R_{\overline{q},n} = success) \wedge (S_{alloc} = First\_Fit)$ **then**
3: $startService(n, \overline{q})$
4: **return** /* The request is hosted on n, SUCCESS */
5: **end if**
6: $N_a \leftarrow \{n^* \in N \cup \{n\}, checkResources(n^*, \overline{q}) = True\}$
7: $N_s \leftarrow$ *nsort*$(S_{alloc}, N_a, \overline{q})$
8: $N_{alloc} \leftarrow \{x : x \in \{N_s[1], \ldots, N_s[i^*]\}, N_s[i^*] = n\}$
9: $l \leftarrow |N_{alloc}|, c \leftarrow 1$
10: **while** $(c \leq l)$ **do**
11: $n_a \leftarrow N_{alloc}[c]$
12: $R_{\overline{q},n_a} \leftarrow testSLA(n_a, \overline{q})$
13: **if** $(R_{\overline{q},n_a} = success)$ **then**
14: **if** $(n_a = n)$ **then**
15: $startService(n_a, \overline{q})$
16: **return** /* The request is hosted on n, SUCCESS */
17: **else**
18: *fwd*$(n_f, S_{fwd}, \overline{q})$
19: **end if**
20: **return**
21: **end if**
22: $c \leftarrow c + 1$
23: **end while**
24: $N_{fwd} \leftarrow$ *nsort*$(S_{fwd}, N, \overline{q})$
25: $l \leftarrow |N_{fwd}|, c \leftarrow 1$
26: **while** $(c \leq l)$ **do**
27: $n_f \leftarrow N_{fwd}[c]$
28: $F_{n_f,\overline{q}} \leftarrow$ *fwd*$(n_f, S_{fwd}, \overline{q})$

29: **if** $(F_{n_f,\overline{q}} = success)$ **then**

30: **return**

31: **end if**

32: $c \leftarrow c + 1$

33: **end while** /* The request cannot be forwarded at all, FAILURE */

The *allocation heuristic*, S_{select}, specifies how a choice has to be made in the case more nodes are valid candidates. The second phase (FWD) is executed if the node is not appropriate for the request, or an SLA could not be signed by the counterparties. In this case, a neighbor will be selected by means of a *forward heuristic* S_{fwd}. Function *fwd*$(x, S_{fwd}, \overline{q})$ routes the request $\overline{q}$ to the node x. S_{fwd} is also passed as parameter, in order to recognize the case on which the strategy belongs to the first group in Table 3, thus the request *is not forwarded* to a node which has been already visited[5]. Function *nsort*$(S, N, \overline{q})$ sorts the node in the set N, returning a set *ordered* on the basis of heuristic S. Function *checkResources*$(x, \overline{q})$ checks whether node x is able to host request $\overline{q}$, i.e. if x belongs to the admissible region; finally function *startService*$(x, \overline{q})$ actually starts the service for the request $\overline{q}$. As discussed in Section 5.7, on which experimental results about resource finding are also presented, procedure described in this section and formalized in Algorithm 1 for Cloud Federations performs well by adopting strategy BFMC (see Table 3).

Table 3: Resource finding forward strategies [BF=Best Fit, WF=Worst Fit, MC=Mass Center, BFMC=Best Fit/Mass Center, MCON=Max Connection, FF=First Fit]

Name	**Use**	**Final selection**
BF	S_{alloc}, S_{fwd}	$x : d(x, q)$ is *minimum* $\forall x \in N(n)$.
WF	S_{alloc}, S_{fwd}	$x : d(x, O = \{0, \ldots, 0\})$ is *maximum* $\forall x \in N(n)$.
MC	S_{fwd}	$x : d(MC(x), q)$ is *minimum* $\forall x \in N(n)$. $MC(x)$ is the Mass Center of x.
BFMC	S_{fwd}	If $d(q, hist_{\overline{q}}(t-1)) \leq d(q, hist_{\overline{q}}(t-2))$ then BF; else MC.
MCON	S_{fwd}	$x : \lvert N(x)\rvert$ is *maximum*, $\forall x \in N$.
FF	S_{alloc}, S_{fwd}	*Random*.

4.4. Trust related issues

As discussed above, resource allocation is performed by means of Algorithm 1, on the base of the establishment of an SLA (*testSLA*), i.e. customer and providers sign an SLA (Service Level Agreement) [3] which contains a clear statement about all the characteristics of the service. Furthermore, as stated in Section 4.1, the topological model of the Hypertrust overlay network is based on the *declared value of*

[5]This special behavior implies that, in such cases, data forwarded from node to node must also include, together with the request, the list of (the *id*s of) the nodes already visited.

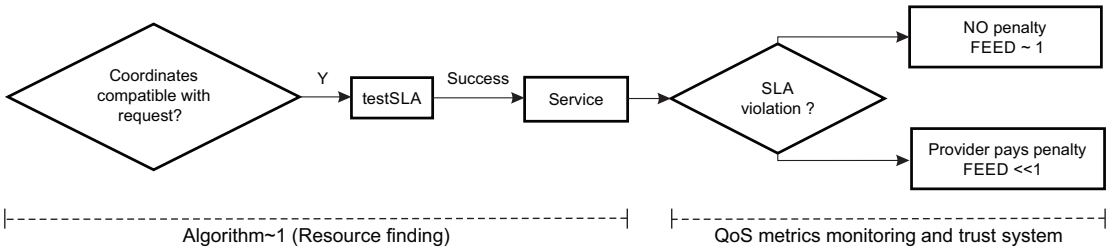


Figure 4: Resource allocation and trust system

service parameters in the $n-$dimensional space. In other words, any nodes which *declare* a number of attributes (i.e. its own coordinates in the hyperspace) compatible with the customer request may decide or not to sign the SLA (i.e. function *testSLA* – line 12 of Algorithm 1 – will give success). Whether or not a node decides to sign the proposed SLA will depend on the business model adopted by the provider [40]. And, whenever the provider performance are not compliant with the constraints specified in the SLA, the customer will give a low value of *feedback* (i.e. $FEED_i(s, j)$, defined in Section 3) with high probability. As a consequence, values of $SR_i(j, c)$ will drop accordingly.

In Figure 4 it is represented a simple schema of the main phases related to the process discussed above: the first one is related to resource finding, whose success is based on the resource attributes declared by the node (and the tendency of the node to take a certain level of risk, i.e. to accept constraints specified in the SLA); next one includes the assessment of SLA violations and assignment of a value $FEED$ by the customer.

A particular concern of the process depicted in Figure 4 is the assignment of the feedback $FEED$. For instance, we may take into account a number of sample parameters listed into table 2, and conceive an example on which customer submits a request for a SaaS containing constraints about *availability*, *response time*. Any of these parameters can assume a different importance in the reliability evaluation when assessing compliance with SLA, e.g. availability might assume a greater weight than response time. As a consequence, the level of reliability assigned by the customer might be related to the weights given to the different QoS parameters. As we explain in Section 5, we model the capacity of the single node to be compliant with the customer request with the ratio between the available resources and those declared, and final QoS is measured accordingly.

5. Evaluation

In this Section we report an experimental evaluation of the proposed approach, organized as follows. Section 5.1 contains the description of the experimental test bed. Section 5.2 contains the results of the simulation of the trust model presented in Section 3; these results involve a scenario on which the the selection of the nodes for services is driven only by the information about trust. It is a first step aimed at performing a preliminary tuning of the relevant parameters of the trust model. It has been useful to

simulate Hypertrust as a whole in a subsequent step. Section 5.3 contains the experimental results that make evidence of the level of resilience of the hypertrust overlay network with respect to the possible shortage of Task Allocators (e.g. due to failures) discussed in Section 2.2. In Section 5.4 the computation of QoS adopted in the experimental test bed is illustrated. Section 5.5 presents the results related to the improvements, in terms of overall QoS, due to the adoption of the Hypertrust model, i.e. the trust-aware decentralized resource finding and allocation discussed in the previous sections. Section 5.6 analyzes the contribution, in terms of QoS, of the Task Allocators and in Section 5.7 discusses the efficiency of the resource finding process, i.e. Algorithm 1 presented in Section 4. Finally, in Section 5.8 we draw our conclusion on the experimental evidences.

5.1. Experimental Test bed

All the simulations were carried by means of a C-based simulation framework which allowed us to simulate the overlay network employed into the Hypertrust system. The simulator was a preliminary version of ComplexSim [44], a C-based general purpose simulation framework specifically designed to simulate complex/P2P systems.

Table 4 reports the main parameters used to set the simulations. We indicate, for convenience, in column 2 (resp. column 5), the section number on which the parameter has been defined (resp. used) for the experiments.

Simulations were performed on a test bed of $50k$ nodes distributed in a bi-dimensional space made by two resource attributes spanning between 0 and $50k$, and 50 different organizations. Requests for services – indicated in row no.2 of Table 4 – were also generated randomly with attributes uniformly distributed between 0 and $50k$, as resource attributes. Nodes have been equally assigned to the organizations which, in turns, provides its own Task Allocator (see Section 2).

Experimental results were obtained by running a variable number of 50-100 simulations with the same configuration. Afterwards, in order to obtain summary data (e.g. median values and outliers) a random set of results was taken into account. As we used a discrete event simulator [44], we performed several simulations with a variable number of epochs, such that a first analysis of results has given the minimum number of epochs from which one could obtain stable results. Since the number of epochs depends on the characteristics of the experiments and we do not consider it as relevant for our purpose, we do not include this information in Table 4.

The nodes behavior was designed in two different manner, depending on the simulation to execute, as indicated on the last row of Table 4. Indeed, as stated in the first part of this section, the results discussed in Section 5.2 rely on the trust model only, while, in order to obtain the results discussed into Sections 5.5 to 5.7, the nodes behavior was designed by combining the execution of the reliability-reputation model discussed in Section 3 and the decentralized resource finding process and overlay network construction and maintenance discussed in Section 4. Finally, nodes acting as Task Allocators were instructed to adopt

the behavior described into Section 3.5. We will make further reference to Table 4 in the following sections as we discuss the experimental results.

	Parameter/concern	**Defined in** Section(s)	**Contribution**	**Value**	**Used in** Section(s)
1	No. of nodes	–	–	50k	5.2-5.7
2	No. of requests	–	–	50k	5.2-5.7
3	No. of organizations (TAs)	2.2	–	50	5.5-5.6
4	No. of service categories	3	–	1	5.5-5.6
5	No. of resource attr.	4	–	2	5.5-5.6
6	Attributes values	4	–	[0,50k]	5.5-5.6
7	Ratio of TA failures	2, 3.5	–	[0.1,0.9]	5.3
8	α, β	3.1, 3.2	SR, RR	V.R. / $0.5, 0.5$	5.2 / 5.5-5.6
9	γ, δ	3.4, 3.5	P, P_{TA}	V.R. / 0.6, 0.5	5.2 / 5.5-5.6
10	$RECC$	3.2, 3.3	R, RR	$= SR$	5.2 / 5.5-5.6
11	$FEED$	3.3	SR, RR	V.R.	5.2, 5.5-5.6
12	Node selection	3, 4	Measured QoS	Trust-based	5.2
13	Node selection	3, 4	Measured QoS	Hypertrust (T) or Algorithm 1 (WT)	5.5
14	Node selection	3, 4	Measured QoS	Hypertrust with and without TAs	5.6
15	Node selection	2	Efficiency	Cases (2.B) and (2.A.1) of Fig. 1	5.7

Table 4: Simulation parameters. V.R.:=variable within a range

5.2. Preliminary evaluation of the trust system

In this section we present the results of a number of experiments aimed at characterizing the behavior of the trust model – discussed in Section 3 – with respect to the parameters α, β, γ. These results allowed us to balance the values of the three parameters above in order to gain maximum benefits. In these simulations, Algorithm 1 (see Section 4) is not used to perform resource finding; instead, the preference of a node for services is based only on the evaluation of data concerning nodes trustworthiness (cf. Section 3), as indicated in line 12 of Table 4.

Parameters involved in these experiments are those cited in rows $1-2, 8-11$ of Table 4. Furthermore, whenever a node i asks a recommendation to a node k about j (cf. Sections 3) we assume that k sends its current value of $SR_i(k)$ (row no.10 in Table 4).

The study was carried as follows:

- a parameter $p \in \{\alpha, \beta, \gamma\}$ assumes value within the range $[0.1 - 0.9]$, and the others two remain fixed around a value of 0.5 (e.g. $\alpha \in [0.1 - 0.9]$ and β, γ fixed);
- feedbacks were sampled by means of three different performance curves:

  lowPerf Low average performance (i.e. feedback) with low variance, $AVG(feed) = 0.3$.

  highPerf High average performance (i.e. feedback) with low variance, $AVG(feed) = 0.9$.

  highVar Variable performance (i.e. feedback) with high variance, $AVG(feed) = 0.5$.

Therefore, the simulations has focused on the generation of feedbacks (which reflect the performances of the nodes) by means of the three cited profiles, which were obtained by sampling values of feedbacks by means of the Beta distribution [63] tuned with appropriate parameters. PDFs (Probability Distribution Functions) of the three curves are reported in Figure 5a. Parameters used for the Beta distribution were tuned with the only purpose to obtain specific PDF, and are not relevant for our purposes.

In particular, the first two performance curves have been conceived in order to take into account situations on which nodes show a level of trustworthiness which is almost the same, without excessive changes. The third one represents a different situation on which nodes may assume variable behaviors.

By means of the three profiles described above, we studied, at first glance, how parameters α and β actually affect the accuracy of computed service reliability (cf. SR expression in Section 3.1) and reputation (cf. R expression in Section 3.2) respectively. The same simulations were performed to characterize the impact of different values of the parameter γ, which is used to balance service reliability SR and reputation R, as discussed in Section 3.4. All the results are presented and discussed below.

• Parameter α. Figure 6a and 6b report the results obtained in the cases *lowPerf* and *HighPerf* described above. In particular, candlestick charts report *minimum*, *First Quartile (FQ)*, *median*, *third quartile* (TQ), and *Maximum* values of the difference $|SR - Feed|$, i.e. the error between the Service Reliability (SR) and the feedback released by the nodes according to several values of parameters α. It can be noted that, when the variance of the feedbacks around the average value is very low, the behavior (in terms of measured reliability) of the nodes is very similar for the nodes of the system, therefore parameter α does not affect the difference $|SR - Feed|$, as expected. Figure 5b reports the results obtained in the case *HighVarPerf*. In this case, (see PDF shown in figure 5a), nodes show a diverse behavior, as sampled feedbacks show high variability. Moreover, setting $\alpha \in [0.4 - 0.6]$ seems to give optimal results, i.e. it allows the nodes to minimize the error $|SR - Feed|$. Indeed, remembering that final preference P (Section 3.4) is computed over Service Reliability (SR) and Reputation (R), then appropriate values of α will allow to be more effective in selecting suitable nodes.

• Parameter β. Similar simulations were performed to study how the parameter β – involved into the computation of Recommendation Reliability RR (cf. Section 3.2) – will affect the accuracy of the

reputation R (cf. Section 3.3) with respect to the effective reliability (i.e. collected feedback). As the results are very similar to those discussed above for parameter α, only the median values of the difference $|R - Feed|$ – where R is the reputation as described in Section 3.2 – are shown by means of figure 7a. Indeed, similarly to the results obtained for parameter α, we observe that it is worth maintaining parameter β in the range $0.3 - 0.7$ (see curve *HighVarPerf*), with a minimum (as for parameter α) around value 0.5.

• Parameter γ. The same experiments were performed on the parameter γ which is used to combine reliability and reputation (see Section 3.4), by measuring the difference $|P - Feed|$. A summary of results (median values) is shown in Figure 7b. As for parameters α and β, results for *HighPerf* and *lowPerf* do not show relevant variation among different values of γ, while the curve *HighVarPerf* reveals that a stable behavior (with minimum error) can be reached with a value of γ between 0.4 and 0.7, with a minimum on the value 0.6.

From this preliminar study we used the setting $\{\alpha, \beta, \gamma\} = \{0.5, 0.5, 0.6\}$ for the simulations presented in Sections 5.5 - 5.6, as reported in lines 8 and 9 of Table 4.

5.3. Resilience in presence of TA shortage

As discussed in Section 2, the first phase of task/service assignment in Hypertrust involves the TA or Task Allocator (cf. Figure 1), which is provided by each organization joining the Hypertrust network. Nevertheless, as stated in Section 2, the absence of a TA (e.g., it is temporary down) can lead the clients to forward their own requests to their own neighbors by means of a simple gossip protocol which is described in Section 2.2.

In order to simulate this particular scenario we selected a gossip protocol provided with a minimal set of features – *cache, Time-To-Live and threshold* – intended to halt the process in a few steps and limit the number of messages. In particular, the threshold – indicated as v – is a probability value: when v has a value close to 1, each gossip message is likely to be propagated to the whole agent's neighborhood, and the *hubs* of the network will generate, in average, an amount of messages which represents an excessive amount of traffic and workload. Therefore, in order to reach most of the nodes, the threshold v should be not too low.

For this study the generated network is identical to that described at the beginning of this section, in particular it is based on parameters listed in rows $1, 3, 5, 7$ of Table 4. A first study of the behavior of the gossip protocol in the Hypertrust overlay network was performed by generating messages from random nodes and eventually tracking the total number of visited nodes (coverage), and the total number of replicated messages (overhead). Results reported into Figure 8 show that setting parameter v in the range $0.5 - 0.6$ and $TTL = 3$ can be enough to reach about the 80% of the nodes (coverage) with an overhead of no more than $3n$ messages.

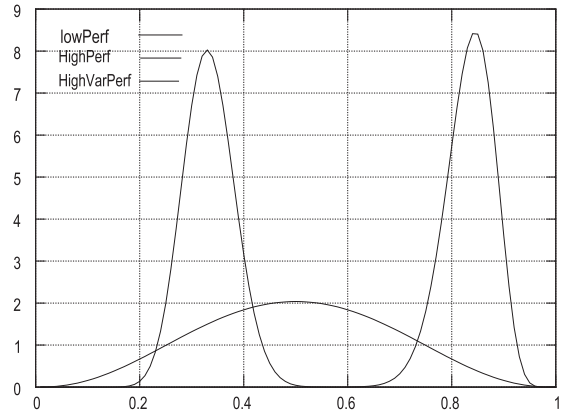
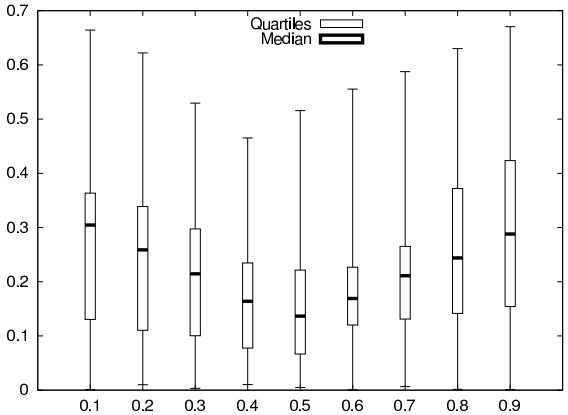


Figure 5: **a)** PDFs of sampled feedbacks



b) α vs $E = |SR - Feed|$ (HighVarPerf)

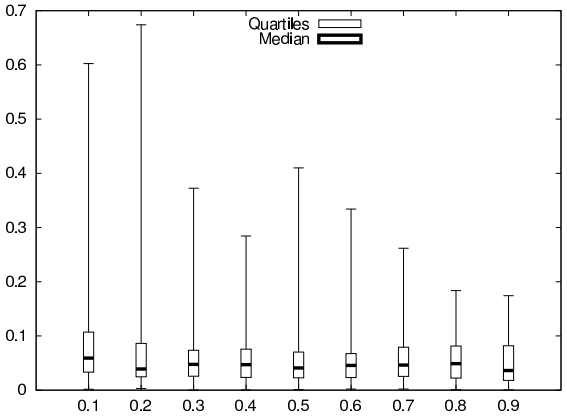
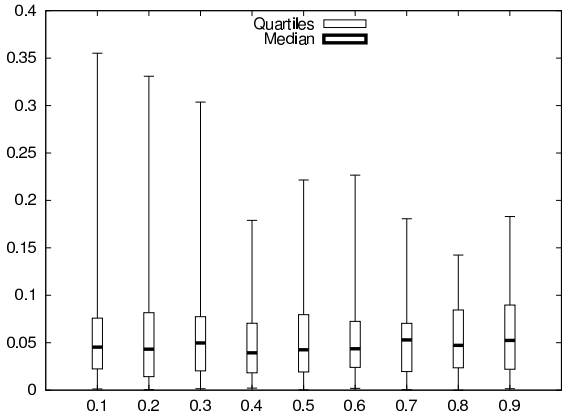


Figure 6: **a)** α vs $E = |SR - Feed|$: (HighPerf)



b) α vs $E = |SR - Feed|$: (LowPerf)

The same experiment was replicated, by including a variable number of TAs failures as reported into line 7 of Table 4, which represents the range of shortage simulated for the TAs. In this case it is interesting to measure the number of times the gossip-based procedure has failed to find another TA in the Hypertrust network. Related results are shown in Figure 9, and make evidence that the impact of TAs failures is minimal (less than 10%) even when there is a shortage of about $50\% - 60\%$ of the TAs, in other words the Hypertrust network shows a good resilience when TA shortages happen.

We remember that, in order to deal with such a failure, in addition to the gossip protocol exploited in Hypertrust , the clients are always able to start the finding algorithm described into Section 4, in order to find a node claiming the requested resources, which is a case similar to the step (2.B), Figure 1.

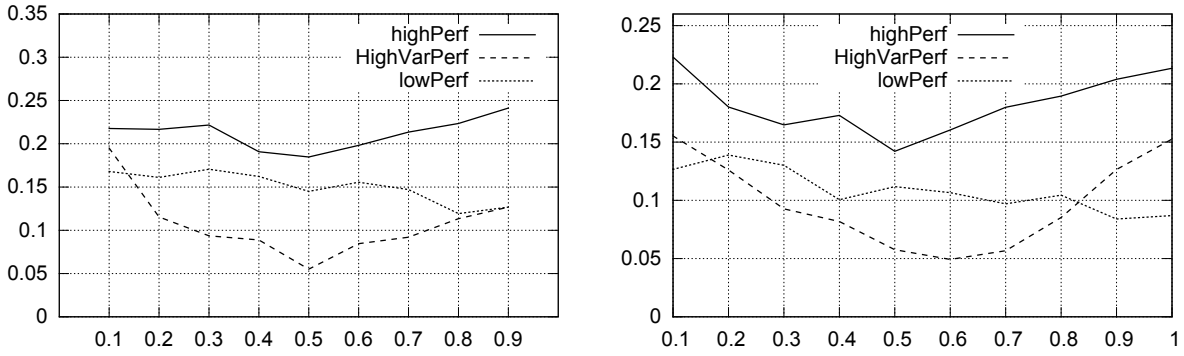


Figure 7: **a)** Parameter β of RR vs Median value of $|R - Feed|$ **b)** γ vs $|P - Feed|$

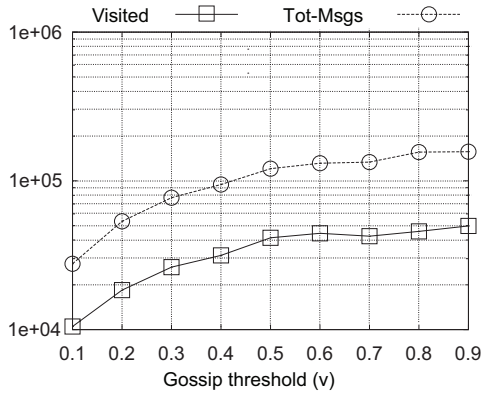


Figure 8: Overhead (Tot-Msgs) and coverage (Visited) of the Gossip protocol

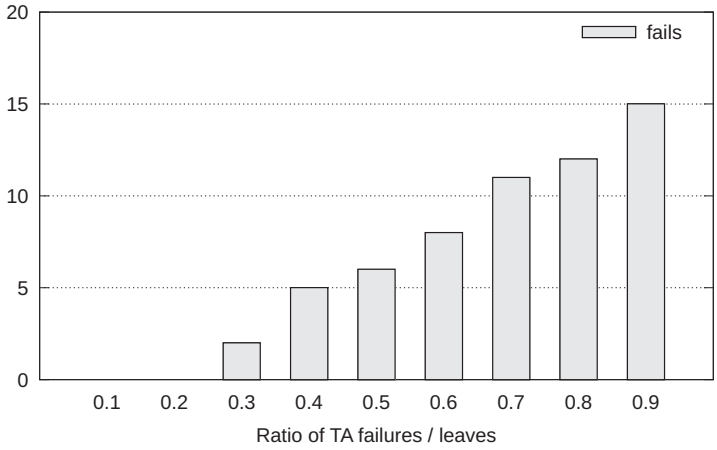


Figure 9: Percentage of fails of the gossip-based protocol (TTL=3) employed to find a TA

5.4. Computation of QoS

As stated at the beginning of this section, the final step of the experimental evaluation is represented by a further set of simulations designed to measure the improvements, in terms of overall QoS, due to the adoption of the Hypertrust model. Related results are presented and discussed into Sections 5.5 and 5.6. Here we explain how we measured the final QoS each time a node has provided a service to another node.

First of all, we modeled the *capacity* of the single node to provide a certain level of quality of service by means of the ratio $Q = q^*/q$, where q^* is the actual amount of resources *hold* by the single node, and q is the amount of *declared* resources. This ratio is closer to 1 when the amount of resources really owned by a node is close to quantity claimed. In fact, ratio Q_i represents the *probability* that node n_i will be reliable in providing a service which needs an amount q of resources.

Since the Hypertrust overlay network is constructed by mapping resource attributes, which may have *qualitative* or *quantitative* nature, we measured the ratio q^*/q as follows. Let be D_j, $(j = 1 \dots l)$ numerical domains for resource attributes which assumes values in numeric ranges (e.g. the computing capability of a CPU in GFLOPS). For convenience we call them *N-domains*. In this case, $q_{ij} > q^*_{ij}$ means that the ith node is declaring more resources that those actually possessed. Let be D_k, $(k = l + 1 \dots n)$ domains for resource attributes which do not take values in numerical ranges. We call them *C-domains*. For instance, in the case "Open Platform" shown into Table 2, a node can offer a subset of different Web APIs over those potentially available. In this case, offering more resources than those actually hold, it means publishing a set of choices greater than those actually available. We indicate as $R_{ij} \subset D_j$ the set of declared resources of node n_i for domain D_j, and $R^*_{ij} \subset D_j$ the actual set of resource attributes hold by the ith node for the same domain D_j.

Given the definitions above, we measure the ratio Q_i as follows:

$$Q_i = \frac{q^*_i}{q_i} = \frac{1}{N}\left(\sum_{j=1}^{l}\frac{q^*_{ij}}{q_{ij}} + \sum_{j=l+1}^{N}\frac{\|R^*_{ij}\|}{\|R_{ij}\|}\right) \tag{1}$$

which refers to a N-dimensional Hypertrust network.

In order to measure the *average QoS* provided to the clients, we distinguished the different cases on which domains are *N-domains* or *C-domains*, as follows:

$$QS(\overline{q}_r, i) = \frac{1}{N}\sum_{j=1}^{N} qs_{rij} \tag{2}$$

$$qs_{rij} = \begin{cases} \min\left(\dfrac{q^*_{ij}}{\overline{q}_{rj}}, 1\right) & 1 \le j \le l, \\ 1 & l+1 \le j \le N \wedge \overline{q}_{rj} \in R^*_{ij}, \\ 0 & l+1 \le j \le N \wedge \overline{q}_{rj} \notin R^*_{ij}. \end{cases} \tag{3}$$

where qs_{rij} is the QoS measured on the resource j provided for the request $\overline{q}_r$ by the node i; $\overline{q}_{rj}$ is the amount of resource j in the request $\overline{q}_r$, q^*_{ij} is the actual amount of resource j hold by the node i. In

particular, the case *C-domain*, $l + 1 \leq j \leq N$, has been included by means of the step function specified in the last two rows of equation 3. Since the trust model discussed in Section 3 is based on the feedbacks released by the users, the computed values of QoS are also used as feedbacks.

5.5. Evaluation of overall QoS

In this section we discuss the results obtained by simulating the dynamics specified by involved in the Hypertrust model as a whole. Here, the trust-based node selection and the overlay network organization and resource finding are employed together (see row no. 13 of Table 4). Resource attributes assigned to the nodes were randomly generated within the range specified in the row no. 6 of Table 4, such that the 50k nodes were uniformly distributed in the overlay network in order to reflect heterogeneity of both resources and services. Furthermore, as specified in Section 5.2, whenever a node i asks an opinion (i.e. a recommendation) to a node k about j, we assume that k sends its current value of $SR_i(k)$ (row no. 10 in Table 4). All the parameters used for these simulations are indicated into rows $1 - 6, 8 - 11$ and 13 of Table 4.

In order to set the level of reliability of cloud resources in the simulated hyperspace, i.e. the ratio $\frac{q^*}{q}$, we considered some studies characterizing performance variability of cloud applications [29, 30]. First of all, results discussed in [29] show that availability is the strength of production cloud services, as it assumes, in average, values very close 100%; this can be daily verified through the various cloud status services provided around the worlds, e.g. CloudHarmony [27]. On the other hand, the same study [29] has shown that performance of common production cloud services is denoted by a significant level of variability, especially in the middle-long term. Variability of performance is due to two main components: the first is mainly due to the sharing of the same physical resource among different VMs [30], while the second is basically due to the real capability of the resources made available by the cloud providers with respect to the requirements dictated by the customers, which can lead to some SLA violations (see Section 4.4). The cited works make evidence of the need – for the customers – to use some "metrics" useful to trigger the switching from a provider to another one, which is the purpose of the trust system presented here. Furthermore, results discussed in [29] show several examples on which median variation of some performance indicators is above 50%. As a consequence, in order to reflect the trend of measures performed in the production clouds, we set the ratio $\frac{q^*}{q}$ to span in the range $0.4 - 1.0$.

The results presented in this section make evidence of the contribution of the trust model presented in this paper. The nodes were categorized into two sets: set *T* (with Trust model) includes a number of nodes instructed to exploit the reliability-reputation model whenever they have to select their collaborators; set *WT* (Without Trust model) includes those nodes which do not use the reputation model nor relative information, and select service providers (or collaborators) basing on the decentralized resource finding algorithm discussed in Section 4. Two kind of tests have been performed, as described below.

By means of the first test, we measured the final QoS with respect to the ratio $Q = q^*/q$ (see Sec-

tion 5.4), and thus on the basis of *how much* the nodes (on average) claims an amount of resources which is different from those actually owned; to this aim, we fixed the percentage of nodes with trust model to 50%, and the ratio Q vary from 0.4 to 1.0, as discussed and motivated above.

During the second set of simulations, we fixed the ratio $Q = q^*/q$ to 0.5 and let the percentage of nodes supported by the trust model varying in the range 20% to 80%. Also in this case, the average QoS was evaluated.

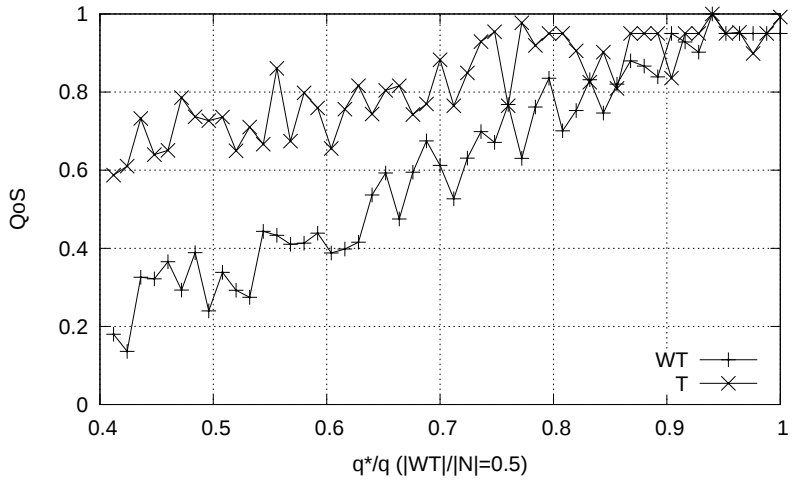


Figure 10: Average QoS for the sets WT (Without Trust) and T (with Trust)

The first set of results is represented in Figure 10 and shows that, on one hand, the nodes using trust information ("T") provide better QoS than the others ("WT"), but, on the other hand, such QoS, in any case, decreases as soon as nodes expose a value different than the real one with a higher probability. Such results mean that the trust model is able to *compensate* the effect of "lies" when a competitive system is considered, but such a compensation is limited to a certain extent. If the discrepancy between resources held and resources advertised is high, nodes could not be able to find proper collaborator and thus to offer the amount of resources requested, with a obvious consequence of a low QoS provisioning.

The said compensation ability of the model is also confirmed by the results provided in Figure 11, which clearly shows that perceived QoS increases as soon as the number of nodes which do not use the trust model decreases.

5.6. QoS contribution of the TA

In order to analyse the contribution, in terms of QoS, due to the TAs (Task Allocators, cf.Section 3.5), we repeated the same experiments reported in Section 5.5. Therefore, these simulations were executed with the same settings, but node's selection was performed in different way, as specified into row no. 14 of Table 4. Indeed, two kind of experiments have been performed: by using and not using the TA to allocate client requests, and related results are depicted in Figure 12. As expected, the "TA" case is very similar to

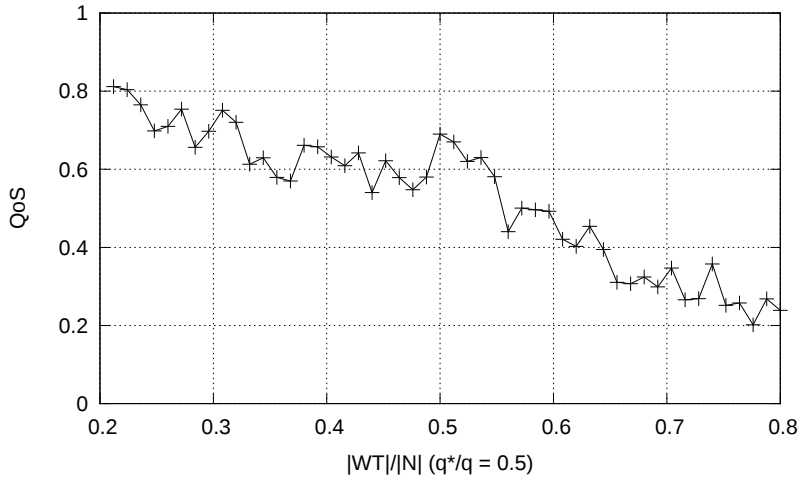


Figure 11: Average QoS vs ratio $|WT|/N$

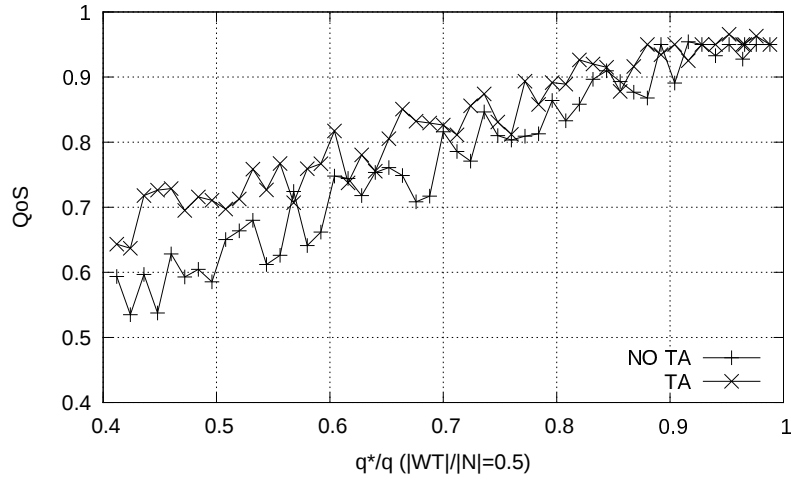


Figure 12: Contribution of the TAs in terms of QoS

the curve labeled "T" of Figure 10, but, more important, it shows a better behavior in term of QoS than the "NO TA", i.e. the provided QoS is lower than in the "TA" case. Hence, the behavior of the system, in terms of provided QoS, improves when TAs are queried for task allocation.

5.7. Decentralized resource finding

In this Section we report some results strictly related to the resource finding process (Algorithm 1 discussed in Section 4) that is employed whenever the node selected by the Task Allocator needs the help by another node, or the TA doesn't hold enough information to select a node for service provisioning. This scenario was discussed into Section 2 and depicted as (2.B) and (2.A.1) in Figure 1. Furthermore, settings of these simulations are basically the same used for experiments of Section 5.5, with the exception of the last block of parameters in Table 4, for which the reader has to refer row no.15.

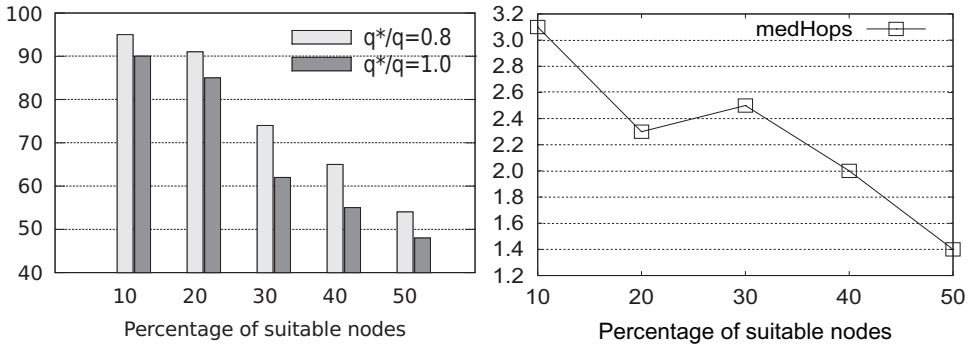


Figure 13: a) Number of times (%) the node selected for task allocation started a resource finding process. b) Number of steps (median value) to find the admissible region.

First of all, we studied the sub-scenario (2.A.1) depicted in Figure 1. In this case the node has been selected by the TA by means of the trust information hold by the TA itself. As a consequence, the node may reside or not into the admissible region and, basing on its capacity to satisfy the request, the node itself may decide to look for a collaborator by the resource finding process explained in Section 4.3. Therefore, the greater the "selectivity" of the request $\overline{q}$ submitted to the TA, the greater the probability that the selected node will start a resource finding process. Figure 13a reports the results of this study in terms of percentage of times the node selected for the task allocation starts a resource finding process to find a set of candidate for a collaboration. It shows that, when the ratio of the node suitable to satisfy the request $\overline{q}$ is about 50%, also the probability that the node starts a resource finding process is around 50%. Differently, when the task is very selective (i.e. the ratio of suitable nodes is around $0.1 \div 0.2$), the probability that the node starts with the resource finding process is very high, as expected. In this case, as specified in Section 2, the trustworthiness of the nodes which reside into the admissible region should be evaluated by (*i*) recommendation or by (*ii*) information already hold by the node which started the resource finding process. In the performed simulations, we tuned the behavior of the nodes to use own trust information as first choice, and, if such information is not available, to ask recommendation to another node. We verified that, as trust information spread among the node of the Hypertrust network, after a brief transitional phase, nodes will ask recommendation only in the 10% of cases, in average. It means that the adopted trust system benefits of the decentralized nature of Hypertrust.

The second issue to consider is the efficiency of the decentralized resource finding process. We verified that the selection heuristic which gives the best results in terms of percentage of failures and average efficiency (i.e. number of steps to find the admissible region) is BFMC (see Section 4.3). As a consequence, in this section we report the median of the number of steps for the given heuristic by means of Figure 13b, observing that is very low even when the percentage of nodes able to satisfy the request $\overline{q}$ is very low.

5.8. Discussion

Hypertrust is a combination of two different models taking into account several different concerns, i.e. trust and efficient decentralized resource finding into large-scale computing infrastructures. The two models have been combined in the scenario depicted into Section 2, and detailed into Section 3 – the trust model – and 4 – the decentralized solution to organize the nodes of the overlay network for efficient resource finding and allocation – such that the experimental results discussed in this section belong to several experimental steps, which correspond to the different scenarios presented into Section 2.

Main parameters of the trust model have been tuned by means of simulations discussed in Section 5, afterwards Hypertrust has been simulated in order to take into account different aspects of the proposed solution. Related results, presented in Sections 5.5-5.7, make evidence that the injection of the proposed trust system into the Hypertrust overlay network will result in major advantages in terms of final QoS given to providers looking for suitable collaborators and final users.

In the comparison between Hypertrust and RRAF [20, 53], which is a trust model based on the combination of reliability and reputation into a unique synthetic trust measure, it can be noted that, in RRAF, the most promising collaborators are selected by analyzing the information about trust among all the nodes of the system. Clearly, this solution can be applied only on a small medium-size system, as stated in Section 1. With Hypertrust the issue above is solved, because it couples different solutions having decentralized nature:

1. The introduction of a variable number of TAs among the several organizations of the federation. Indeed, Task allocators are able to collect past experiences (i.e. reliability), allowing actors (clients or provider) looking for suitable service providers to exploit these information in a decentralized manner. The effectiveness of this approach has been highlighted by experimental results shown into Section 5.6, while the non-functional aspect of resilience with respect to the shortage of TAs, have been proved into Section 5.3. Task Allocators represent, in fact, a hierarchical solution which helps requesters to select suitable nodes in the less "complicated" cases, as remarked in the next point.
2. As TAs rely only on reliability information coming from past experiences within their own organizations, these information might be not enough to provide good suggestions. In Hypertrust several additional chances are given to the customer and/or any service provider looking for collaborators. Indeed, by approaching the admissible region by means of the decentralized Algorithm 1 (Section 4), the analysis of trustworthiness of the nodes can be limited only to those nodes of the federation. The efficiency of this approach, in terms of average number of steps needed to approach the admissible region, has been described into Section 5.7. These additional features clearly exploit the fully decentralized features of the Hypertrust overlay network.

From the observations above it can be stated that Hypertrust outperforms the centralized, trust-based solution based on the combination of reliability and reputation (e.g. RRAF). We will discuss some other decentralized trust systems in Section 6.

The nature of Hypertrust allows requesters to find nodes having suitable attributes with high efficiency and fault tolerance, as proved in Section 5.7. Authors of [15] have presented a two-layered gossip based protocol to support a greedy resource finding process similar to that developed for Hypertrust. The two layers are built by means of a random overlay network providing network connectivity, and an additional, structured network to perform resource finding with minimum overhead. Decentralized resource finding approach of Hypertrust is different because the overlay construction does not rely on additional networks. Indeed, as explained in [42] and in Section 4, the "essentially critical neighbors" are preserved, such that the network will never disconnect and the average minimum path is maintained low. Resource finding approach is different due to the different structure of the network. In particular, although the overhead related to the path necessary to approach the target, shown in [15], is comparable with that of Hypertrust, discussed in Section 5.7, the topology of Hypertrust reflects the resource attributes of the nodes, allowing the set of nodes compatible with the request to be explored with a low overhead. In other words, this feature is important because is coupled with the features of the trust system, which benefits from the structured organization of the Hypertrust network, as stated in the previous point 2.

6. Related Work

Trust models [26, 47, 50, 57] allow to exploit information derived by direct experiences (reliability) and/or opinions of others (reputation) to trust potential partners by means of a single measure [2, 28, 54, 68]. Trust measures will be as accurate as greater the number of reputation information is [7], even though virtual environments encourage malicious behaviors.

Distributed trust systems have been studied since 1990s. The main contribution of [70] was to clarify that formal reasoning about trust in distributed systems is not possible and, consequently, made distinction between direct experiences (reliability in Hypertrust) and recommendation.

Ntropi [1] continued to study trust in distributed systems by proposing a framework for building trust protocols which do not rely on third parties (Trust Authorities). Trust information are propagated by means of interactions between agents that, in order to adjust trust measures, provide to rate their past experiences. They basically introduced the concept of trustworthiness of recommenders, which, in Hypertrust, we measured as recommendation reliability.

EigenTrust [33] deals with the problem of identifying inauthentic files in a Peer-to-Peer (P2P) file sharing network in order to decrease the number of downloads. Each peer build its *local trust* representation by assigning a rate to any other peers, while a *global trust* is built by exploiting a transitivity criteria for which peers honest about files they have uploaded, will be also honest when providing recommendations. Conversely, in Hypertrust the trust model we measure recommendation reliability (RR). Since the Hypertrust scenario is different from P2P file sharing, we can't assume that honesty in characterizing services is equal to honesty in providing recommendation, although they are related concepts.

A distributed system handling uncertainty and variability is presented in [23]. The authors propose a trust model to deal with the discrepancy between the information provided by other agents, and experience is used to anticipate the behavior of the other (i.e. the "quality" of that actions) agents. In this way, the agent is able to adapt more effectively its behavior to changes in the environment for its own benefit. This approach is due to the considerations that different agents may use a different framework to represent and reason about things. Nevertheless, in Hypertrust we categorized services so that we assume that trust agents assisting users, providers and TAs agree on a common meaning of, e.g. availability and set of criteria to evaluate it. On the other hand, the assumption above results from the fact that computing infrastructures are federated.

The multidimensional aspect of trust in a cooperative agent systems was investigated in many studies. For instance, in REGRET [55] the aggregation of multiple faces of reputation, i.e. information coming from witnesses, social relations, roles and general properties, is adopted. A similar agent system is FIRE [28], which deals with open, benevolent and honest agents by considering direct experiences, relationships, witnesses and certified reputations. Such multiple sources give versatility to FIRE but require to tune many parameters. In Hypertrust the concept of multidimensionality is used to construct the overlay network, which eventually supports the decentralized resource finding. Once the admissible region for the requested resources has been found, trust information are exploited to select the best node.

In general, all the cited trust systems use both reliability and reputation measures, similarly to Hypertrust, to trust agents in different contexts by combining such measures. Some of them *i*) weight opinions provided by other agents by means of their trust, *ii*) adopt a "mentor", *iii*) use discrepancies between computed trust and observed behaviors or *iv*) exploit a trust measure specifically designed for a particular scenario. Finally, all the presented systems, similarly to Hypertrust, do not use cryptographic techniques.

In the context of cloud services, the Cloud Security Alliance[6] —an international organization for promoting best-practices for security assurance in cloud computing— issued a specification called *Cloud Control Matrix*[14], which defines a set of assets and principles that cloud computing provides should met to design *secure* services. It is somewhat a trust verification *by-design*, which, while begin sound and meaningful, does not ensure that, at run-time, the desired trust level is always kept. Moreover, it focuses more on security aspects rather than reliability, as instead in our approach.

The approach used for decentralized resource finding presented in this work has been studied by the authors in [41, 42, 43]. Similar studies [35, 59] aimed at characterizing how a typical "greedy search" based on local information is able to find the short path with high probability on scale free and small world networks respectively. Hypertrust overlay network is based on exploiting local information, and provides the desired convergence in a few steps, as shown into the experimental section, proving also that the network remains connected over time. Moreover the resource finding over the Hypertrust overlay

[6] `http://cloudsecurityalliance.org`

network has been tested on highly dynamic environments, i.e. nodes change their attributes as allocations have been performed. Heuristics provided to optimize the finding show that resource finding still performs well in dynamic conditions or very limited resources.

A similar area of interests includes some works [4, 8, 16, 17, 22, 39, 49] aiming at using different economic models for trading resources on large-scale distributed systems for different application domains, including CPU cycles and storage space. In particular, in [39] a market-based approach to computational grid resource management is presented. As in the work discussed in this paper, the authors of [39] consider a competitive scenario, where multiple Grid resource agents sell resources and multiple task agents ask to buy the resources. Two optimization problem are addressed by a price-directed market-based algorithm to solve the grid task agent allocation problem. In particular, the second class of addressed problems take into account a QoS constraint over the time needed to complete a sequence of tasks, and the experiments show that the performance are better of the classical round-robin allocation. In the proposal in [22] the authors try to take into account the nature the decisional process of federated Cloud providers which have to look for the trade off between in-sourcing and out-sourcing of virtual computing infrastructures. By defining a number of equations which take into account revenues and costs, they define the way on which local schedulers can take their decision considered as optimal.

In [58] the authors present an algorithm of fully decentralized resource discovery in Grid computing at large scale. Their algorithm is based on a simply unicast request (provided with a TTL) with the addition of a reservation algorithm which allows users to bring forward the resource discovery mechanism to find more resources. They evaluated the performance of the algorithms by comparing with the first-found-first-served algorithm, showing comparable performances. First of all, they do not deal with trust related issues; moreover, their algorithm is based on a unicast message, while Hypertrust is based on the construction of an overlay network showing the characteristics of a small world network.

In [71] the problem of resource allocation in cloud federations is addressed, with focus on the typical scenario on which providers have the need of dynamically allocate resources across multiple cloud environments by means of an inter-trust relationship architectural model. Nevertheless, there are two main issues in their work that differ with Hypertrust. First of all, they deal only with the problem of verifying the identity of providers, by envisioning the role of a number of IdP (Identity Provider) [11, 12] to support inter cloud collaborations. IdP trust indexes exploited from their past behavior, in terms of reliability. Conversely, Hypertrust focuses on the mapping of resources in order to obtain an overlay network which is exploited to perform an efficient, fully decentralized resource finding process. Furthermore, Hypertrust contains a complete reliability/reputation model to support the providers when suitable collaborators have to be selected.

7. Conclusions and future work

In this paper we discussed a decentralized solution to support trusted resource finding and allocation into competitive, large scale, federations of utility computing infrastructure. Our approach, like other similar techniques developed in the past for competitive distributed multi agent systems, allows a node to choose the most promising collaborators, based on both a direct trust measure (reliability) and a reputation measure derived from the recommendations of the other nodes. However, unlike most of the past proposals, which generally need to explore the whole agent space for selecting collaborators, our technique exploits a decentralized technique which organizes the servers/computing nodes in an overlay network featuring given characteristics.

More specifically, Hypertrust allows any user to exploit an efficient finding process of the available resources, giving to each node the opportunity to use its trust model for limiting the search of collaborators to an admissible region previously discovered by the decentralized resource finding technique. The basic model provides a decentralised procedure to construct an overlay network including all the nodes of the federation, by exploiting the resource status similarity. In other words, peers featuring a similar amount of resource availability tend to be interconnected by means of the links of the overlay thus forming clusters which, in turn, are connected together by means of few long links. An algorithm has been designed to exploit the main characteristics of the Hypertrust overlay network, on which nearest nodes are preferred for link organizations, such that *communities* or *clusters of similar nodes* will feature a short intra-cluster distance. Moreover, *long links* between clusters are kept, allowing the system to preserve its connectivity.

An extensive set of experiments has clearly shown how the nodes which use our trust-based approach significantly outperform the nodes that do not use any trust model to select their collaborators. This validates the approach, although the trust-based selection of collaborators is not performed in the whole space of nodes. Moreover, we also introduced a special node, the Task Allocator and, although the maintaining of a TA is not mandatory for the organizations, this special node allows the user to delegate the selection process for the task, in such a way improving the QoS. Some experiments have shown that the adoption of the TAs does not prevent the distributed nature of the approach but the overall QoS is, in the average, improved.

For the future, we are planning to extend the trust model in order to exploit meaning and correlation of specific, critical trust concerns, e.g. security, privacy, accountability, auditability [36]. Indeed, Hypertrust model relies on reliability and recommendation measures relating to actors (i.e. providers) and kind of services. Mapping also specific concerns into the trust system, along with a framework for evaluating these concerns, would provide a finer support for customers and providers looking for business partners.

Acknowledgement

This work has been supported by project PRISMA PON04a2 A/F funded by the Italian Ministry of University and by the NeCS Laboratory - Dep. DICEAM - University Mediterranea of Reggio Calabria.

References

[1] A. Abdul-Rahman, S. Hailes, A distributed trust model, in: Proc. of New Security Paradigms, Work. on, ACM, 1997, pp. 48–60.

[2] K. Aberer, D. Z., Managing trust in peer-2-peer information systems, in: Proc. of Information and Knowledge Management, 10th Int. Conf. on, ACM, 2001, pp. 310–317.

[3] M. Alhamad, T. Dillon, E. Chang, Conceptual sla framework for cloud computing, in: Digital Ecosystems and Technologies (DEST), 2010 4th IEEE Int. Conf. on, IEEE, 2010, pp. 606–610.

[4] Y. Amir, B. Awerbuch, A. Barak, R. S. Borgstrom, A. Keren, An opportunity cost approach for job assignment in a scalable computing cluster, Parallel and Distributed Systems, IEEE Transactions on 11 (7) (2000) 760–768.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Comm. of the ACM 53 (4) (2010) 50–58.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, ACM SIGOPS Operating Systems Review 37 (5) (2003) 164–177.

[7] A. Birk, Boosting cooperation by evolving trust, Applied Artificial Intell. 14 (8) (2000) 769–784.

[8] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in grid computing, Concurrency and computation: practice and experience 14 (13-15) (2002) 1507–1542.

[9] R. Buyya, R. Ranjan, R. N. Calheiros, Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, in: Algorithms and architectures for parallel processing, Springer, 2010, pp. 13–31.

[10] J. Cao, K. Hwang, K. Li, A. Y. Zomaya, Optimal multiserver configuration for profit maximization in cloud computing, Parallel and Distributed Systems, IEEE Transactions on 24 (6) (2013) 1087–1096.

[11] A. Celesti, F. Tusa, M. Villari, A. Puliafito, How to enhance cloud architectures to enable cross-federation, in: Cloud Computing (CLOUD), 2010 IEEE 3rd Int. Conf. on, IEEE, 2010, pp. 337–345.

[12] A. Celesti, F. Tusa, M. Villari, A. Puliafito, Three-phase cross-cloud federation model: The cloud sso authentication, in: Advances in Future Internet, 2010 2nd Int. Conf. on, IEEE, 2010, pp. 94–101.

[13] M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, D. Zamboni, Cloud security is not (just) virtualization security: a short paper, in: Proc. of the 2009 ACM work. on Cloud computing security, ACM, 2009, pp. 97–102.

[14] Cloud Security Alliance, Cloud control matrix (2015).
URL `https://cloudsecurityalliance.org/research/ccm/`

[15] P. Costa, J. Napper, G. Pierre, M. van Steen, Autonomous resource selection for decentralized utility computing, in: Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE Int. Conf. on, IEEE, 2009, pp. 561–570.

[16] A. Di Stefano, C. Santoro, A marketplace-based economical model for resource provisioning agreement in a P2P grid, in: Proc. of *HPDC Work. on the Use of P2P, GRID and Agents for the Development of Content Distribution Networks (UPGRADE 2005)*, Paris, France, 2006.

[17] A. Di Stefano, C. Santoro, An economical model for resource management in a grid-based content distribution network, Future Generation Computer Systems 24 (3).

[18] I. Foster, C. Kesselman, The grid: blueprint for a new computing infrastructure, Morgan Kaufmann, 2004.

[19] B. Furht, A. Escalante, Handbook of Cloud Computing, Springer, 2010.

[20] S. Garruzzo, D. Rosaci, Roles of reliability and reputation in competitive multi agent systems, in: Proc. of the On the Move to Meaningful Internet Systems: Confederated Int. Conf. OTM 2010, Part I, LNCS, Springer-Verlag, 2010, pp. 439–442.

[21] D. Gefen, E. Karahanna, D. W. Straub, Trust and tam in online shopping: An integrated model, MIS quarterly 27 (1) (2003) 51–90.

[22] Í. Goiri, J. Guitart, J. Torres, Economic model of a cloud provider operating in a federated cloud, Information Systems Frontiers 14 (4) (2012) 827–843.

[23] M. Gómez, J. Carbó, C. Benac-Earle, An anticipatory trust model for open distributed systems, vol. 4250 of LNAI, Springer-Verlag, 2007, pp. 307–324.

[24] T. Grandison, M. Sloman, Trust management tools for internet applications, in: Proc. Trust management (iTrust'03), 1st Int. Conf. on, Springer-Verlag, 2003, pp. 91–107.

[25] R. L. Grossman, The case for cloud computing, IT professional 11 (2) (2009) 23–27.

[26] Y. Han, A survey of trust and reputation management systems in wireless communications, Proc. of the IEEE 98 (10) (2009) 291–298.

[27] C. Harmony, Cloudharmony. com, february 2012.

[28] T. Huynh, N. Jennings, N. Shadbolt, An integrated trust and reputation model for open multi-agent system, Autonmous Agent and Multi Agent Systems 13.

[29] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, D. H. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, Parallel and Distributed Systems, IEEE Transactions on 22 (6) (2011) 931–945.

[30] A. Iosup, N. Yigitbasi, D. Epema, On the performance variability of production cloud services, in: Cluster, Cloud and Grid Computing, 2011 11th IEEE/ACM Int. Symp. on, IEEE, 2011, pp. 104–113.

[31] M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks, ACM Transaction on Computer Systems. 23 (3) (2005) 219–252.

[32] A. Jula, E. Sundararajan, Z. Othman, Cloud computing service composition: A systematic literature review, Expert Systems with Applications 41 (8) (2014) 3809–3824.

[33] S. Kamvar, M. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in: Proc. of World Wide Web, 12th Int. Conf. on, ACM, 2003, pp. 640–651.

[34] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, KVM: the linux virtual machine monitor, in: Proc. of the Linux Symp., vol. 1, 2007, pp. 225–230.

[35] J. M. Kleinberg, Navigation in a small world, Nature 406 (6798) (2000) 845–845.

[36] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, B. S. Lee, Trustcloud: A framework for accountability and trust in cloud computing, in: Services (SERVICES), 2011 IEEE World Congress on, IEEE, 2011, pp. 584–588.

[37] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, Software: Practice and Experience 32 (2) (2002) 135–164.

[38] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, M. Kunze, Cloud federation, in: CLOUD COMPUTING 2011, The 2nd Int. Conf. on Cloud Computing, GRIDs, and Virtualization, 2011, pp. 32–38.

[39] C. Li, L. Li, Competitive proportional resource allocation policy for computational grid, Future Generation Computer Systems 20 (6) (2004) 1041 – 1054.

[40] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computingthe business perspective, Decision Support Systems 51 (1) (2011) 176–189.

[41] F. Messina, G. Pappalardo, C. Santoro, Hygra: A decentralized protocol for resource discovery and job allocation in large computational grids, in: Proc. of Computers and Communications ISCC, IEEE Symp. on, IEEE, 2010, pp. 817–823.

[42] F. Messina, G. Pappalardo, C. Santoro, Exploiting the small-world effect for resource finding in P2P grids/clouds, in: Proc. of Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 20th IEEE Int. Work., IEEE, 2011, pp. 122–127.

[43] F. Messina, G. Pappalardo, C. Santoro, Decentralised resource finding in cloud/grid computing environments: A performance evaluation, in: Proc. of Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 21th IEEE Int. Work., IEEE, 2012, pp. 143–148.

[44] F. Messina, G. Pappalardo, C. Santoro, Complexsim: a flexible simulation platform for complex systems, International Journal of Simulation and Process Modelling 8 (4) (2013) 202–211.

[45] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, Algorithmic game theory, vol. 1, Cambridge University Press Cambridge, 2007.

[46] D. Niyato, A. V. Vasilakos, Z. Kun, Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach, in: Proc. of the 2011 11th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing, IEEE Computer Society, 2011, pp. 215–224.

[47] S. Ramchurn, D. Huynh, N. Jennings, Trust in multi-agent systems, Knowledge Engeenering Review 19 (1) (2004) 1–25.

[48] R. Ranjan, A. Harwood, R. Buyya, et al., Grid federation: An economy based, scalable distributed resource management system for large-scale resource coupling, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia.

[49] O. Regev, N. Nisan, The popcorn market. online markets for computational resources, Decision Support Systems 28 (1) (2000) 177–189.

[50] P. Resnick, R. Zeckhauser, F. E., K. Kuwabara, Reputation systems, Comm. of ACM 43 (12) (2000) 45–48.

[51] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, et al., The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 4–1.

[52] L. Rodero-Merino, E. Caron, A. Muresan, F. Desprez, Using clouds to scale grid resources: An economic model, Future Generation Computer Systems 28 (4) (2012) 633–646.

[53] D. Rosaci, Trust measures for competitive agents, Knowledge-Based System 28 (2012) 38–46.

[54] D. Rosaci, G. M.L. Sarnè, S. Garruzzo, Integrating trust measures in multiagent systems, International Journal of Intelligent Systems 27 (1) (2012) 1–15.

[55] J. Sabater, C. Sierra, Reputation in gregarious societies, in: Proc. of Autonomous Agents, 5th Int. Conf. on, ACM, 2001, pp. 194–195.

[56] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artificial Intelligence Review 24 (1) (2005) 33–60.

[57] J. Sabater-Mir, M. Paolucci, On representation and aggregation of social evaluations in computational trust and reputation models, International Journal of Approximate Reasoning 46 (3) (2007) 458–483.

[58] S. Tangpongprasit, T. Katagiri, K. Kise, H. Honda, T. Yuba, A time-to-live based reservation algorithm on fully decentralized resource discovery in grid computing, Parallel Computing 31 (6) (2005) 529–543.

[59] H. Thadakamalla, R. Albert, S. Kumara, Search in spatial scale-free networks, New Journal of Physics 9 (6) (2007) 190.

[60] T. Thanakornworakij, R. Nassar, C. B. Leangsuksun, M. Paun, An economic model for maximizing profit of a cloud service provider, in: Availability, Reliability and Security (ARES), 2012 7th Int. Conf. on, IEEE, 2012, pp. 274–279.

[61] A. N. Toosi, R. N. Calheiros, R. Buyya, Interconnected cloud computing environments: Challenges, taxonomy, and survey, ACM Computing Surveys (CSUR) 47 (1) (2014) 7.

[62] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, R. Buyya, Resource provisioning policies to increase iaas provider's profit in a federated cloud environment, in: High Performance Computing and Communications (HPCC), 2011 IEEE 13th Int. Conf. on, IEEE, 2011, pp. 279–287.

[63] C. Walck, Handbook on statistical distributions for experimentalists (2007).

[64] W. Wang, B. Li, B. Liang, Towards optimal capacity segmentation with hybrid cloud pricing, in: Distributed Computing Systems (ICDCS), 2012 IEEE 32nd Int. Conf. on, IEEE, 2012, pp. 425–434.

[65] Y. Wang, X. Lin, M. Pedram, A game theoretic framework of sla-based resource allocation for competitive cloud service providers, in: Green Technologies Conf., 2014 6th Annual IEEE, IEEE, 2014, pp. 37–43.

[66] D. Watts, S. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.

[67] L. Wu, S. K. Garg, R. Buyya, C. Chen, S. Versteeg, Automated sla negotiation framework for cloud computing, in: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM Int. Symp. on, IEEE, 2013, pp. 235–244.

[68] L. Xiong, L. Liu, Supporting reputation-based trust for peer-to-peer electronic communities, IEEE Transaction on Knowledge and Data Engineering 16 (7) (2004) 843–857.

[69] H. Xu, B. Li, Maximizing revenue with dynamic cloud pricing: The infinite horizon case, in: Communications (ICC), 2012 IEEE Int. Conf. on, IEEE, 2012, pp. 2929–2933.

[70] R. Yahalom, B. Klein, T. Beth, Trust relationships in secure systems-a distributed authentication perspective, in: Research in Security and Privacy. Proc., IEEE Symp. on, IEEE, 1993, pp. 150–164.

[71] K.-H. Yeh, An efficient resource allocation framework for cloud federations, Information Technology And Control 44 (1) (2015) 64–76.

[72] Q. Zhang, L. Cheng, R. Boutab, Cloud computing: State-of-the-Art and research challenges, J. of Internet Services and Applications 1 (2010) 7–18.