A Distributed Reputation System for Super-Peer Desktop Grids

Peter Merz, Florian Kolter, Matthias Priebe
Distributed Algorithms Group
Department of Computer Science
University of Kaiserslautern
D-67663 Kaiserslautern, Germany

Email: {pmerz,f_kolter,priebe}@informatik.uni-kl.de

***Abstract*—Desktop Grids leverage otherwise unused resources of idle desktop computers, providing vast amounts of cumulative computational power. However, resource sharing in Peer-to-Peer environments with selfish participants suffers from the free-riding phenomenon unless the environment provides appropriate countermeasures. In Peer-to-Peer-based Desktop Grids, cooperative participants require protection against free-riding job distributors. In this article, we present a decentralized shared-history reputation mechanism designed for use with Desktop Grids built on dynamic super-peer structures. Embedded in a distributed Desktop Grid workflow model, our concept promotes reciprocity and discourages free-riding. In simulations based on real-world network delay and workload information, we show that our concept offers a considerable speedup over non-distributed computation while effectively thwarting free-riding and maintaining the system's commitment to robustness and scalability.**

***Keywords*—Peer-to-Peer; Distributed Computing; Desktop Grids; Free-Riding; Reputation Systems**

I. Introduction

For decades, supercomputers have been the method of choice in cases where the solution to a problem depended on the availability of massive computational power. They provide computational resources to an extent that exceeds that of ordinary desktop computers by orders of magnitude. However, supercomputers are costly to build and operate, and there may be considerable lead time until one becomes available.

On the other hand, desktop PCs are occasionally idle, spending unused processor cycles for no return. This is especially true for multi-core CPUs in which one or more cores may find themselves underloaded for considerable periods of time. Harvesting these idle cycles to form a virtual supercomputer that offers its cumulative computation power to the general public represents a promising alternative to conventional supercomputing. Inside this virtual computation engine, a self-organizing system would interconnect the participating machines, splitting large jobs into small tasks, distributing the tasks among the desktop machines, and reassembling the individual results into a single one as it would have been computed had the job been processed by a supercomputer in the traditional way. As in Grid computing, resources are connected via wide-area networks, in particular the Internet. A major difference lies in the way resources are deployed and administrated. In Grid Computing, resources are supervised by trustworthy administrators and have been acquired with the intention of being remotely accessed to handle distributed work. They are commonly available for long periods of time.

On the contrary, desktop PCs operated by scientific institutions, companies, public authorities, and private households are purchased for the purpose of performing work locally. A desktop computer is usually less powerful than a computational Grid resource, and the quality of its Internet connection may vary considerably [2]. The remote utilization of idle cycles from these devices has begun as volunteer computing. In volunteer computing, computer owners are asked to voluntarily donate idle CPU time. In this model, a user would run problem-specific software that receives a job of a well-known kind from a pre-defined server, computes it in its idle periods, and returns the results to the server. A prominent example of this approach is SETI@home [3] which analyzes radio signals from outer space for hints of extraterrestrial life.

For a number of applications, Peer-to-Peer (P2P) networks have emerged as a viable alternative to client/server schemes such as SETI@home. In their fully distributed form, P2P networks contain no single point of failure, scale to millions of participants – called *peers* due to the absence of a hierarchy –, and include properties of self-organization that improve their robustness. Distributed computing in a P2P setting enables every participant to initiate the computation of a job by asking its fellow peers to join the computing effort. Because of its focus on desktop resources, a system that taps the idle computation resources provided by desktop units, is based on a P2P network and enables everyone to submit arbitrary jobs for distributed computation is referred to as a Desktop Grid [4], [5], [6], [7], [8]. The cumulative potential leveraged by a Desktop Grid grows with the number of resources connected to it. Hence, a Desktop Grid benefits from the capability to integrate a heterogeneous range of computers in a dynamic, volatile, decentralized environment. The requirement to support a substantial number of participants emphasizes the importance of scalability in this context. The construction of Desktop Grid structures on P2P overlays improves scalability in large-scale settings [5], [9]. It is further improved by the concept of *super-peers* which combines advantageous properties of client/server systems and unstructured P2P topologies [10]. Unlike ordinary Grids, peers participating in Desktop Grids bring all the resources including the infrastructure. Since overlay dynamics are explicitly accounted for, joining a Desktop Grid does not require significant administrative efforts besides running the software which is required for access.

While a Desktop Grid offers attractive features, the peers are operated by humans whose goals may be selfish rather than altruistic. In the purely selfish case, a peer may join the Desktop Grid, submit its job, wait for its completion, and once finished, leave the system. In this case, that particular peer does not contribute any resources to the system but exploits it for its own good: it is a *free-rider*. While a certain amount of free-riders might be tolerated, the system would break down when there are too many because the demand for resources exceeds the supply. Hence, a Desktop Grid needs to deploy effective measures to thwart free-riding.

In this article, we propose a reputation system tailored for distributed Desktop Grids that are built on super-peer structures. This reputation system mitigates the adverse effects of selfish, uncooperative peer behavior while maintaining the commitment to meet scalability and robustness requirements. It permits cooperative peers to detect and eventually avoid free-riding job distributors. It is accompanied by a workflow model which adapts the common distributed computing workflow to a super-peer setting. By means of experiments, we demonstrate the gains over configurations with no protection for cooperative peers. This article is an extended version of [1], adding an investigation of the collusion phenomenon and a series of experiments that quantify the overhead incurred by the proposed reputation system, the numbers of accepted and declined task requests, and the effects of varying free-rider fractions. Moreover, it contains enhanced coverage of reputation-related issues in P2P networks and a summary of aspects relevant to the construction of super-peer topologies.

The remainder of this article is organized as follows. Section II sketches the architecture of super-peer overlay topologies along with a distributed algorithm that creates and maintains a topology of this type. Section III outlines issues caused by rational-selfish peer behavior. Section IV presents the proposed Desktop Grid model which consists of workflow, reputation, and incentive components. Section V describes the scenario, setup and outcome of simulation experiments with the proposed model. Section VI discusses related efforts. The article concludes with directions for future research in Section VII.

II. Overlay

Super-peers are peers that act as relays for a number of attached common peers called *edge peers*, while at the same time, they form a network of equals among themselves. That way, the entire P2P network may remain connected over a substantially smaller number of links than before [10], [11]. Besides enhancing scalability, super-peers may route communication between edge peers that cannot directly communicate with each other due to the presence of firewalls. An example for this feature is depicted in Figure 1 where edge peer p_1 wishes to communicate with edge peer p_{14}. For the first hop, p_1 sends the message to its super-peer, c_1, which forwards it to the destination's super-peer, c_4. Finally, c_4 passes the message on to the destination peer, p_{14}. In a fully meshed super-peer overlay as shown in Figure 1, the network diameter is only 3 hops.

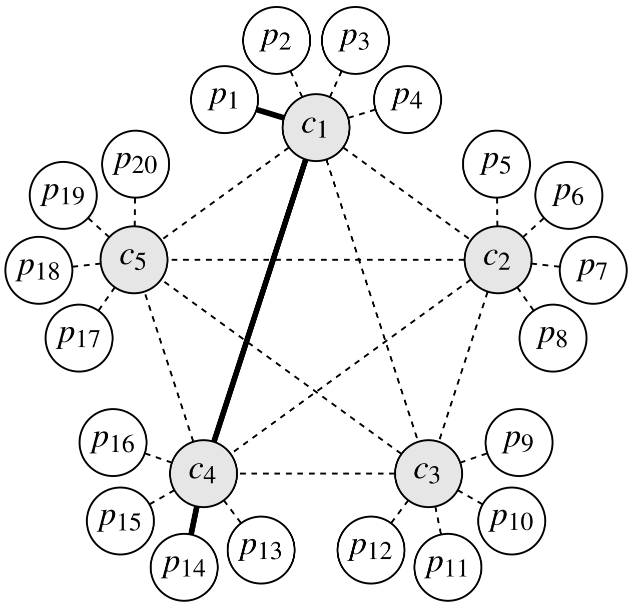

Fig. 1. Example of a P2P network with selected super-peers

This section briefly describes the Super-Peer Selection Algorithm (SPSA) [12], [13], a distributed algorithm that creates and maintains a delay-optimized, self-organizing, fully meshed super-peer overlay. SPSA has been conceived to establish a P2P overlay on top of which a Desktop Grid can be built. The algorithm runs continuously on every peer in the overlay. It promotes suitable peers to super-peer level and makes nearby edge peers connect to them. No central instance is required for SPSA to perform. The first peer to join the overlay instantly becomes a super-peer, while all peers joining subsequently receive edge peer status at join time.

SPSA strives to minimize the number of connections a super-peer needs to keep up. In a fully meshed overlay, a super-peer maintains one connection to each of its edge peers and one to every other super-peer, respectively. Let C be the set of super-peers. With n peers in the system, the number of connections per super-peer is minimized for $|C| = \sqrt{n}$ [13]. Every super-peer can estimate the current number of peers in the overlay by counting the number of connections held to other super-peers; let this number be h. Also, every super-peer knows the number of edge peers attached to it; let this number be z. Now, a super-peer changes its role and downgrades to edge peer level if $z < 0.5 \cdot h$. Conversely, if a super-peer finds itself overloaded by determining that $z > 2 \cdot h$ holds, it picks one of its edge peers and promotes it to super-peer level so that some of the remaining edge peers switch to the new super-peer. To minimize the end-to-end message delay, edge peers always connect to the closest super-peer, overloaded super-peers always pick the edge peer for promotion that minimizes the sum of delays regarding its connections, and all super-peers regularly check if one of their edge peers can perform their duty at lower cost in terms of network delay. This approach scales well when delays are estimated using network coordinates [14], [15].

III. Peer behavior

A free-rider is an individual who does not provide any resources to a system but exploits it selfishly [16], [17], [18], [19], [20], [21]. While an altruistic setting may support a certain extent of free-riding, it is expected to collapse if exceedingly many peers choose to act as free-riders. Therefore, Desktop Grids benefit from mechanisms which curb free-riding. One important aspect in this context is *reputation*. Reputation conveys information about a peer's past conduct to influence expectations of its future behavior [22]. It quantifies the risk in trusting others. By disseminating aggregated information on peer behavior, a reputation scheme encourages selfish peers to act in a fair manner [17]. That way, peers may avoid cooperation with other peers that exhibit poor reputation, providing an incentive to peers to accumulate positive feedback by acting accordingly. Generally, reputation aggregates feedback information. Without a reputation mechanism, a peer's abusive behavior toward a certain peer will not harm the misbehaving peer when interacting with other peers.

It is assumed that every peer has a private type which determines its strategy. The model distinguishes between three types of peers [16], [23], [24], [25], [26]:

- *Altruists* serve to the best of their capabilities every request issued by other peers without asking for reciprocity. Hence, exhibiting the opposite behavior to complete selfishness, altruists always obey and cooperate with other peers regardless of external rewards. Thus, altruists require no protection against free-riders.
- Selfish participants expect to receive a benefit from the Desktop Grid:
  - *Collaborators* are generally willing to cooperate if they can expect their counterpart to eventually reciprocate. In particular, they are ready to reserve resources for use by others provided that these resources are not occupied by free-riding users.
  - *Free-riders* are selfish participants that do not wish to contribute resources to the Desktop Grid; rather, they strive to exploit the computational resources of unsuspecting or unprotected peers.

Collaborators not protected by a reputation system cannot tell about the past behavior of peers they are about to interact with. Hence, their resources are easily occupied by free-riders that exploit this lack of protection. Technically, while both accept any job, altruists and unprotected collaborators differ in that altruists do not mind being exploited because they gain utility from the mere act of unconditionally donating their resources to others [25]. On the contrary, collaborators seek reciprocity and shun exploitation. A reputation system enables collaborators to assess the risk in dealing with other peers, creating the opportunity to choose whether to engage in or to abstain from an interaction with a given remote peer, and generating incentives for peers to adopt reciprocity in their own behavior.

The interaction between two peers is commonly modelled as an instance of the *Iterated Prisoner's Dilemma* where both players can pick one of the strategies *cooperate* or *defect* (i.e. attempt to exploit the other player) independently from each other [23], [26], [27]. The situation depicts a conflict between group rationality and individual rationality: two rational players will make the game result in the game's only Nash equilibrium (defect, defect) which is individually rational but not Pareto efficient since both would enjoy more benefit if they had played (cooperate, cooperate). Regarding the modelled types, altruists will always choose to cooperate and free-riders will always choose to defect. Hence, two altruists will cooperate, two free-riders will not cooperate, and a free-rider will exploit the resources of an altruist but neither will mind. On the contrary, collaborators seek other collaborators (or alternatively, altruists): they are willing to cooperate if the other player cooperates, too. In the original Prisoner's Dilemma or when the number of dilemma iterations is known beforehand, defection is the rational strategy to play. However, if the number of played rounds is unknown, cooperation becomes the rational strategy. If rational players cannot estimate the negative consequences of defection, they are better off by choosing cooperation [27].

In P2P networks, peers may obtain non-persistent identities at virtually no cost, leading to the whitewashing phenomenon in which a selfish peer is tempted to leave and immediately rejoin the network with a new identity after having performed actions that are detrimental to its reputation in the first place [19], [20]. With no persistent identities, there is no definitive means to distinguish a true newcomer from a whitewasher. Placing general mistrust in newcomers is one way to discourage whitewashing [5], [22]. The cost of this approach has been assessed in [22]. An alternative to general mistrust has been suggested in [19]. It consists of an adaptive stranger policy that judges new peers on the basis of new peers' behavior in the past by maintaining a *stranger account* that accumulates recent experience with new nodes. Plenty of free-riders joining the Desktop Grid will tend to decrease a stranger account's value, while a majority of collaborators will increase it. Hence, a stranger account may serve as a predictor for an unknown peer's reliability.

In this article, we propose to combine the adaptive stranger policy with a super-peer-based approach. Super-peer overlays consist of a comparatively small number of super-peers and a large number of edge peers. At join time, a peer has a neutral reputation with all other peers, influenced only through the stranger account. When super-peers regularly charge their edge peers reputation for the super-peer service and spread this reputation information to other peers, an edge peer's reputation declines unless the edge peer performs work for other peers or becomes a super-peer itself. However, to become a super-peer, the SPSA protocol requires an edge peer to be appointed by another super-peer. This way, a super-peer is free to choose a trustworthy edge peer for promotion. We expect that with this scheme, a reputation system will effectively repulse free-riders in the long term.

IV. Concept

The absence of a central authority leads to the issue of effective resource discovery. Nodes will need to communicate with their fellow peers to keep track of available resources, handle job submissions and care for their completion. This joint effort may be directed by a Desktop Grid middleware. When run on every participating machine, it enables peers to coordinate their actions, providing a platform for applications to have arbitrary jobs processed by the Desktop Grid. In the concept presented in this article, every peer can submit a job which will be split into a number of tasks that may be processed by other peers. A peer that processes a task will be referred to as *worker*, whereas a peer that submits a job will be referred to as *initiator* of that job. Every peer is supposed to first compute a local power index based on a dummy job which may be distributed along with a Desktop Grid middleware. The computation of that job yields a benchmark number that eases the comparison of the peers' capabilities, similar to the *Horse Power Factor* in [28].

Given a super-peer network in which every edge peer is attached to exactly one super-peer each, our approach operates in a distributed way and completely bypasses the need for a virtual currency by relying on a reputation-based mechanism. There is no single point of failure and no requirement for pre-trusted nodes. Design goals have also included scalability and churn resistance to enable the concept to prevail in a dynamic environment. We ignore irrational malicious peers (i. e., those who cause harm to the system without benefitting from such action). To ensure fairness, cooperative peers grant reputation to other peers for successfully completing a task and for acting as a super-peer. The remainder of this section introduces all components of our concept including the workflow model, the proposed reputation system and incentives for peers to collaborate.

A. Workflow

The workflow model assumes that initiators want to lease computation time on idle workers. An initiator generally wishes to minimize its job's total time to completion (referred to as *makespan* [7]). Following this approach, the time the slowest worker takes will determine the makespan.

Two kinds of jobs are considered. The first kind is *independent-task applications* (ITA), also known as *bag-of-tasks* [5], where the workers process their tasks independently and isolated from each other. Prominent examples for ITA jobs include parameter-sweep applications [7]. Moreover, there are *connected problems* (CP) [28] which benefit from the interaction of workers, requiring the participating workers to perform their computation efforts concurrently. For example, connected problems include evolutionary algorithms that address combinatorial optimization problems. There, workers communicate with others to indicate that they have found a new best known solution. That way, the best known solution is replicated among the workers, removing the strict need to replace a failing worker. Since the global optimum is unknown, a criterion to terminate the computation is required, e. g., a deadline. In the ITA case, however, a job is split into a number of tasks that are distributed among the workers. It is unknown how long a task will take to be completed by a worker, and a worker's failure implies that its assigned task must be computed again by another worker.

A potential initiator first determines its job requirements which encompass the minimum and maximum numbers of desired worker peers, a worker's minimum acceptable local power index, the job type (CP or ITA) and a deadline until which all workers must have finished their computations. If the initiator is an edge peer, it passes these requirements on to its super-peer. The super-peer broadcasts a message containing a request for worker participation and the job requirements to all other super-peers, which, in turn, forward it to their attached edge peers. Conversely, the super-peers collect approvals from their edge peers and forward them to the initiator along with details on their respective power index and availability. This multiplexing reduces network load by preventing peers from individually replying to the initiator. Now, the initiator holds a list of worker candidates which it sorts according to power index, reputation and availability, and picks a sufficient number of workers which shall perform the distributed computation. The workers start computing upon reception of code and data transmitted to them by the initiator.

It is assumed that tasks running on worker peers can contact the initiator anytime. That way, a task may return its result after completion to the initiator, and may even submit intermediate results as proof-of-work to confirm that the worker has indeed provided the task with computational resources. The results are returned directly by the task running on the worker machine. A dishonest worker needs to know the result message's format to fake its contents, requiring it either to tap the line and intercept the message sent by the running task or to retrieve the message format from reengineering the task's code. Both ways entail efforts that exceed the effort of computing the actual task. Hence, a rational peer will choose to honestly compute the task. The workflow ends with the initiator validating the partial results and merging them into the complete outcome that would have resulted had the job been computed on a single machine in a non-distributed way.

B. Reputation system

A reputation system collects information on the past behavior of other peers [29], [30]. It enables cooperative peers to identify other cooperative peers, and it creates incentives for peers to act cooperatively by providing rewards for cooperative behavior [29], [31]. It enables casting the "shadow of the future" [27] which ensures that peers need to consider the consequences of defection. Cooperative peers learn about the true willingness of others to reciprocate only when requesting remote resources on their own, but they can reduce the risk of erroneously donating resources to uncooperative peers by relying on a reputation system.

The model presented in this article has been built on the notion that the contribution of computational resources for use by others deserves rewards. These rewards are implemented by positive feedback. In a Desktop Grid setting, peers prefer to cooperate with other peers that have high reputation, and they wish to establish and maintain the potential to become successful initiators. This requires the ability to attract a sufficient number of workers to have the job computed in a reasonable period of time. Since the level of attraction is directly linked to their individual reputation, peers benefit from behaving in a way that earns them positive reputation. The aim of the proposed reputation system is to enable cooperative peers to detect free-riding initiators, shielding the detecting peer from being exploited. The system is fully distributed: every collaborating peer runs an instance of it. The proposed model assumes that free-riders will not compute any tasks for others; when collaborators refuse to compute tasks for free-riders, free-riders are left with the choice to switch to cooperative behavior or to leave the Desktop Grid.

Reputation systems differ in the way reputation is recorded. If a peer records reputation information only on those peers with which it has directly interacted in the past, the recorded information is unforgeable, but in large dynamic systems with considerable turnover, peers rarely encounter the same peer again for interaction. In contrast, shared-history reputation systems additionally incorporate experiences made by other peers [19].

Reputation may also serve as a currency replacement. Workers charge initiators a payment for the work they have performed on behalf of the respective initiator; the payment depends on the work's volume and is deducted from the initiator's local reputation account held by the worker. This way, reputation may be built by serving others and spent by straining others.

In the context of Desktop Grids, spoof feedback, the transmission of faulty or purposefully forged opinions on other peers, is the most prominent kind of attack on shared-history reputation systems [32]. In particular, a shared-history reputation system may face *collusion*, the phenomenon of interacting selfish peers (*colluders*) that mutually forge reputation ratings to benefit fellow peers in the colluder group and harm those outside it. In the context of this article, a colluder always attempts to free-ride.

Due to its purposeful, organized and durable nature, collusion is a severe form of spoof feedback. If peers act selfishly but autonomously, there is no incentive to spread counterfeit reputation information on other peers. However, if a group of colluding peers agrees to spread only positive information on the group's members to other peers, ordinary peers need to remain vigilant when receiving shared-history feedback from a remote peer. Systems in which all peers agree on a common (objective) reputation for every peer are especially affected by collusion [19].

In a super-peer scenario, colluders may become super-peers. In this state, they may exert malevolent influence on their edge peers. For instance, a super-peer which participates in a collusion could decide to forward only messages from fellow colluders and drop all others, or exchange forged reputation with its edge peers. To assess threats of this kind, one may consider the worst case that all free-riders know each other and form a single collusion group. Let n be the number of peers in the overlay out of which $\sqrt{n}$ are super-peers as selected by SPSA. Moreover, let p be the probability that an arbitrary peer is a colluding free-rider. The number of super-peers which belong to the colluder group follows the binomial distribution with parameters $\sqrt{n}$ and p. Let X be a random variable with this distribution. Its distribution function is [33]

$$Pr(X \le x) = F(x) = \sum_{i=0}^{x} \binom{\lceil \sqrt{n} \rceil}{i} \cdot p^i \cdot (1-p)^{\lceil \sqrt{n} \rceil - i} \qquad (1)$$

Hence, the probability that at least one of the super-peers is a colluding free-rider amounts to

$$Pr(X \ge 1) = 1 - Pr(X \le 0) = 1 - (1-p)^{\lceil \sqrt{n} \rceil} \qquad (2)$$

For a network of $n = 100$ peers and $p = 0.1$, this probability equals 65.13 %; at $p = 0.2$, it reaches 89.26 % and at $p = 0.3$, 97.18 %. To attain a probability of 50 % that a colluder is among the super-peers, the fraction of free-riders in a 100-peer system need only be $p = 0.067$, i. e., 7 colluding peers are already sufficient. In a 10 000-peer system, it is $p = 0.007$, requiring 70 colluders to achieve the same effect. These results show that it is feasible for a comparatively small number of colluding peers to get hold of a part of a super-peer overlay's infrastructure.

The impact of collusive behavior on P2P overlays becomes still more severe with the *Sybil attack* which constitutes a particularly intense special case of collusion [34]. It may occur in systems where identities cannot be verified due to the lack of a trustworthy authority. In the Sybil scenario, a group of colluding peers may be controlled by the same entity. Hence, the assumption of administrative autonomy and independence of peers does not hold for the Sybil attack. There is no general solution to this attack in fully distributed systems that satisfies the requirements of efficiency and scalability, but a number of approaches exist to limit the attack's effects in various application domains [35]. In Desktop Grids, besides weighting remote feedback with the submitting peer's credibility [19], a similarity measure can alleviate the effects of spoof feedback [31], [36]. Using a similarity measure, peers pay more attention to peers reporting similar opinions as one's own.

Peers spread feedback on the behavior of other peers they have interacted with. In particular, this concerns the initiator-worker relationship. For this purpose, every peer maintains a list of recent contacts, i. e., peers with which successful interaction has taken place in the past. Additionally, peers maintain reputation records on arbitrary other peers by keeping a list of weighted positive (a) and weighted negative (b) experiences per remote peer. From these parameters, a peer may set up a β-distributed random variable B(a,b) for every remote peer and use its mean to reflect the respective remote peer's trustworthiness. While raw feedback is binary (0 equals poor, 1 equals good performance), reputation may assume any value $v \in [0,1]$ due to feedback weighting and aggregation. Both a and b will be initialized with 1 such that the distribution mean $\frac{a}{a+b}$ equals the neutral reputation value, 0.5, in the beginning [37]. This may be overridden by the stranger account.

To keep the reputation list size manageable, there shall be a time window that will remove outdated peers (i. e., those with no recent sign of activity). The list is split in two: one direct interaction history and one holding shared-history information received from other peers. Reputation is periodically spread to the last batch of peers with which a peer has interacted in the past (for every edge peer this includes its respective superpeer). Stored reputation is subject to a periodically applied exponential decay using a weighting factor $z \in (0,1)$ to focus on the rated peer's recent behavior, effectively leading to a short-term history [19], [37]. Essentially, for every remote peer on which reputation information is available, the decay process performs the following update:

$$a := 1 + (a-1)\cdot z \tag{3}$$

$$b := 1 + (b-1)\cdot z \tag{4}$$

That way, past reputation will fade out over time, taking the distribution's parameters gradually back to the initial setting unless new reputation information arrives.

When a request for participation as a worker from an initiator i with reputation r_{ji} arrives at a peer j, j will check with its reputation list if i is acceptable. Peer j instantly agrees to become a worker for i if j's directly observed reputation about i exceeds 0.5, or j's combined weighted local and remote observations about i reach or exceed the neutral value of 0.5. The neutral value's inclusion at this point supports bootstrapping the system in the beginning when no worker-initiator interaction has taken place yet.

If i is rejected according to the aforementioned rule, j switches to random-acceptance mode: i's reputation with j is compared to the moving average m of the last f seen initiators incorporating a tolerance factor $\beta \in [0,1]$. If $r_{ji} > (1+\beta)\cdot m$, i is accepted, while i is rejected if $r_{ji} \leq (1-\beta)\cdot m$. If $(1-\beta)\cdot m \leq r_{ji} < (1+\beta)\cdot m$, j makes its decision on a randomized basis by drawing a random number $d \in [0,1]$ from a uniform distribution. If $d < r_{ji}$, i is accepted, otherwise it is rejected. The random-acceptance mode supports the emergence of cooperation in environments with many free-riders. With a modest probability, it permits unknown collaborating peers to be accepted as initiators. However, the higher the tolerance level, the more free-riders are likely to pass through. It is for this reason that β needs to be chosen carefully to avoid an adverse impact on the reputation system's effectiveness. In all cases where no reputation information is available on an initiator, the worker resorts to the stranger account.

Every peer i maintains an interval t_i which defines the frequency of its reputation exchange. This interval need not equal the reputation decay interval. After a period of length t_i has elapsed, i transmits its stored reputation information (the weighted average of the distribution means for both the local and the remote observations) to the q members of its recent contacts list. Since reputation information basically only consists of one peer ID and one floating point number per rated peer, the reputation-related message volume remains reasonable. Peer i expects its contacted peers to reciprocate by replying with their respective reputation information. If a contacted remote peer fails to fulfill this expectation multiple times in a row, i removes it from its contact list. That way, the reputation exchange cycle encourages mutual updates.

When j receives a reputation list from i, j first checks i's credibility. Peer i's information will be incorporated into j's remote reputation list if i's reputation with j, r_{ji}, or the *personalized similarity measure* [36] between i and j, quantified as s_{ji}, exceeds the neutral value of 0.5. If i passes this test, j proceeds with normalizing the received reputation items (*votes*) to avoid subjective exaggerations, otherwise i's information is discarded. In the positive case, let v_{ik} be the vote cast by i about peer k. For every vote v_{ik} submitted by i, j now computes

$$v^*_{ik} = \frac{v_{ik} - 0.5}{\sum_l |v_{il} - 0.5|} \tag{5}$$

to normalize i's information, yielding $v^*_{ik} \in [-1,1]$. Let

$$r^*_{ji} = \alpha + \max\{0,\ 2\cdot(1-\alpha)\cdot(r_{ji} - 0.5)\} \tag{6}$$

with $\alpha \in [0,1]$ being the minimum weight that j applies to remote opinions. $r^*_{ji} \in [\alpha,1]$ expresses the particular weight which j associates with i's opinion. Let g_{ji} be that weight weighted with the similarity between i and j, i. e., $g_{ji} := s_{ji} \cdot r^*_{ji}$, to yield the final weight j applies to each of i's votes. Ultimately, for every v^*_{ik}, j updates its remote observation list: if $v^*_{ik} > 0$, i's opinion on k is positive, hence j sets $a_{jk} := a_{jk} + g_{ji}\cdot v^*_{ik}$. If $v^*_{ik} < 0$, i's opinion on k is negative, so j sets $b_{jk} := b_{jk} + g_{ji}\cdot v^*_{ik}$.

Local reputation is modified as the outcome of a worker-initiator relationship. Workers receive rewards from initiators for completed tasks only. Hence, workers will want to finish an ongoing computation. An initiator i determines a worker's reputation gain primarily by the computation time the worker has expended. Concisely, let w be the worker in question, t the number of i's reputation periods – including fractions – of length t_i the task has occupied computational resources on w, and power$_x$ the local power index of peer x. Now, i quantifies w's work effort h_{iw} as

$$h_{iw} = \frac{\text{power}_w}{\text{power}_i}\cdot t \tag{7}$$

Peer i recomputes the β-distribution parameters it keeps for w, a_{iw} and b_{iw}. If w has completed its task, i sets $a_{iw} := a_{iw} + h_{iw}$, otherwise $b_{iw} := b_{iw} + 2\cdot h_{iw}$, as a direct observation. The factor of 2 serves as a penalty for the failed completion of a task. Note that neither i's choice of its reputation update interval length t_i nor the worker-initiator power ratio biases the reputation propagation as reputation information transmitted to other peers is processed in an aggregated form only (by means of the distribution mean), independent of both the interval length and the power ratio.

Workers charge initiators reputation as the price of computation by adding negative feedback to the respective initiator's local reputation account. Since free-riders do not accrue positive reputation on their own, subsequent requests for workers by free-riders will be declined.

C. Incentives

We wish to obtain a self-sustaining system with a mutual reputation exchange. To this end, we propose a reputation-based incentive scheme that operates without the need to exchange virtual money. Rather, it fosters reciprocity in a dynamic environment. This scheme is built on the notion of positive reputation that may be obtained by adequate behavior. Positive reputation can only be earned by accomplishing work, boosting the willingness of peers to become workers. The following actions supplement the model's incentive scheme:

- Workers and initiators have established a trustful relationship after having successfully cooperated. Afterwards, they begin to exchange reputation information periodically. In our model, successfully completing a worker-initiator cycle or becoming part of a super-peer-edge-peer relationship are the only ways to exchange reputation information with another peer. If either peer cheats, the other peer will consider the relationship not to be a success. However, both peers are interested in reputation information about other peers, and successfully interacting over a longer period of time builds mutual trust. This hampers collusion as any peer wanting to spread false feedback will first need to complete a worker-initiator cycle or become a super-peer. Moreover, peers will stop transmitting reputation information to other peers if these remote peers do not provide reputation information themselves. This is similar to the Tit-for-Tat reputation exchange strategy used in [32]. We suggest the reputation exchange between two peers to be alternately initiated. This prevents peers from cheating by replying with the same reputation values it has just received; it also prevents the similarity measure from being tricked, because with equal reputation ratings, the similarity is maximum (1).
- Super-peer activity is considered a computation job with unknown duration. All edge peers are considered initiators and their respective super-peer is the single worker of their job. Consequently, super-peers charge their edge peers reputation while edge peers grant their super-peers reputation. Peers can receive an additional incentive to become super-peers by granting a super-peer the right to require its attached edge peers to compute a task for it. Every edge peer takes its turn in a round-robin or randomized fashion, and the turn length equals the length of the super-peer's reputation exchange interval. That way, every edge peer expends some computation power to the super-peer once in a while. If a super-peer attempts to cheat by requiring one edge peer too often to compute a task, that edge peer may disconnect at any time and seek another super-peer. Conversely, if an edge peer refuses to compute, the super-peer itself disconnects the link. The job may concern the computation of an arbitrary problem, or in the default case, be a dummy job which returns proof-of-work messages to the super-peer.

With these incentives, it becomes attractive for peers to become super-peers and to exchange reputation information with others.

V. Experiments

A reputation system faces the challenge of distinguishing cooperative from free-riding peers. In the beginning, all peers consider all other peers equal due to lack of experience. With time passing, collaborators interact with other collaborators. Hence, the proposed reputation system is expected to adapt to its environment over the course of time, accepting a limited number of free-riders in the beginning but repelling them after having swung in. To assess the performance of our approach, a number of experiments have been conducted in a custom discrete-event simulation environment using Java [12]. We were particularly interested in the speedup attained by our Desktop Grid concept compared to single-machine computation and in its resistance to free-riders. We also wanted to determine the impact of free-riding behavior on unprepared, unprotected Desktop Grids. We have expected the average makespan of jobs to increase with the number of free-riders. In our experiments, the makespan computation has exclusively concerned jobs initiated by cooperative peers. The makespan experienced by free-riders has not been taken into account as the utility of free-riders is considered irrelevant in this context.

A. Setup

Initiators are anticipated to submit computationally intensive jobs only since small jobs may be computed locally without needing to deal with the additional complexity of a Desktop Grid [18]. Hence, a Desktop Grid is likely to be exposed to jobs that would otherwise be processed by supercomputer-class systems. To model the volume of such jobs, we have resorted to real-world workload traces[1] of jobs processed by a Linux cluster at the Ohio Supercomputing Center. After data sanitization, the data of 30414 jobs were available. In a simulation experiment, jobs were either all CP or all ITA; in the ITA case, the initiator did not know about the length of the job, so every ITA job was actually distributed. Since ITA jobs tend to exert more negative influence on the average makespan than CP jobs due to the unknown runtime, ITA jobs have been chosen unless noted otherwise. Cooperative peers initiated jobs independently from their current local load. The interarrival time between two jobs submitted by the same cooperative

[1] from http://www.cs.huji.ac.il/labs/parallel/workload/l_osc/index.html

peer was modelled as a uniformly distributed random variable $U(0; 5 \cdot \text{median single-machine CPU time per job})$.

The sample network consisted of 100 nodes with known RTT between all nodes as measured on PlanetLab [38], [39]. The maximum delay was estimated at 3 seconds. The simulation environment included a SPSA super-peer overlay optimization process running on every peer which reshaped the overlay for low end-to-end communication delay.

The reputation spreading scheme pushed reputation information to the elements of its recent contacts list (workers, initiators, and, in case of an edge peer, its super-peer) every 45 seconds of simulated time. The reputation aggregation mechanism placed a weight of 0.75 on local and a weight of 0.25 on remote observations. The history decay factor z was set to $z = 0.999$ to enable a graceful decay and long keeping of positive reputation accrued through previous work for other peers. To limit the load, the recent contacts list has been restricted to $q = 10$ elements and the unique initiator reputation history also to $f = 10$ elements. The minimum granted reputation weight α was set to 0.1, the random-acceptance tolerance level β to 0.02. After having completed a task, worker j modified an initiator i's local reputation by setting $a_{ji} := a_{ji} + \frac{1}{4} \cdot t_c$ and $b_{ji} := b_{ji} + \frac{3}{4} \cdot t_c$, where t_c is the number of reputation intervals – including fractions – that j has spent computing i's task.

Free-riding behavior has been modelled as the refusal to accept foreign tasks for local computation, the desire to submit a job for distributed computation upon entry, and the departure from the system when the job has been completed. Once a free-rider had left the system, a very brief period of simulated time (3 seconds) elapsed until it was replaced by another free-rider which immediately attempted to submit a new job. This aggressive behavior partially suppressed the establishment of reputation-backed cooperative relationships and spread mistrust in the system due to the large amount of jobs initiated by free-riders; it strained the reputation system, testing a case which is substantially more severe than is likely to be encountered in practice due to the exceedingly high volume of jobs submitted by free-riders. Free-riders attempted 10 times to distribute their job before giving up on that job. Additionally, in some experiments, free-riders could collude. Collusion has been modelled on the basis of the wish to spread reputation information which maximally benefits all members of the colluding peer group while at the same time, maximally debases all other peers to reduce their capability to initiate a job of their own. Hence, a colluder's feedback on fellow colluders was quantified as 1, while feedback on other peers was quantified as 0. We have followed a worst-case evaluation approach, assuming that all free-riders in the system are colluders, and there is only one colluder group in the system, i. e., every colluder knows all other colluders. In our system, the only way to transmit forged reputation information to a collaborating peer is through the super-peer-edge-peer relationship. Since super-peers and edge peers consider themselves workers and initiators, respectively, they exchange reputation information until the colluding peer has completed its job and leaves the overlay.

While our model supports heterogeneous worker capabilities, we have assumed identical peer equipment in these experiments to facilitate comparisons. The willingness of peers to process tasks has been limited to a maximum of 24 hours of computation time per task. The simulation lasted for 7 days of simulated time. The admission of jobs has been narrowed to include only those of 10 days or less of total computation time on a single CPU. Moreover, jobs with very short CPU time requirements (less than 5 minutes) have been removed because such jobs are unlikely to be distributed on a Desktop Grid. In total, 18 156 jobs have met the admission requirements and as such, have been made available to the simulation environment. The longest admitted job required 8 days and 5 hours on a single CPU, the shortest one 5 minutes. Median job CPU time consumption equalled 5 hours, 48 minutes. For every experimental scenario, 30 runs have been performed.

B. Results

We have first examined the impact of free-riding behavior on a Desktop Grid with cooperative workers not protected by a reputation scheme. To this end, we have exposed the system to free-rider fractions of various sizes and measured the average makespan experienced by cooperative peers. Figure 2 shows the outcome, confirming that the makespan grows with the fraction of free-riders. This reflects the negative impact of free-riders on system performance, highlighting the necessity to introduce countermeasures that inhibit free-riding.

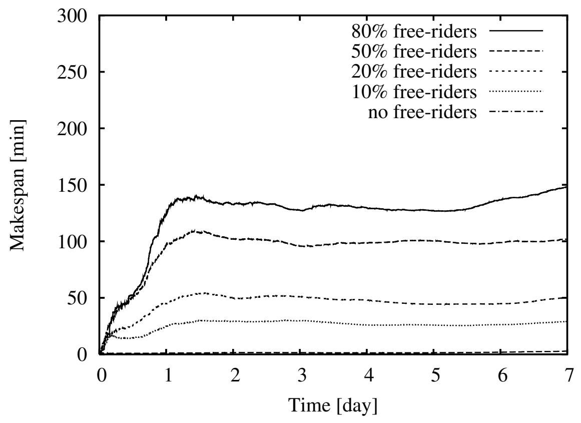

Fig. 2. Effects of free-riding on average makespan

In another experiment, the performance of a free-riding-resilient Desktop Grid which uses the proposed reputation system has been compared to an unprotected but otherwise identical Desktop Grid. Figures 3, 4, 5 and 6 show the outcome for free-rider levels of 10 %, 30 %, 50 % and 70 %, respectively. The makespan is considerably lower when a reputation scheme is available to the collaborating peers than in the original setting where unprotected collaborating peers cannot detect free-riders. With free-riders in the system, the reputation system takes time in the beginning to adapt to the environment but then swings into stabilizing the average

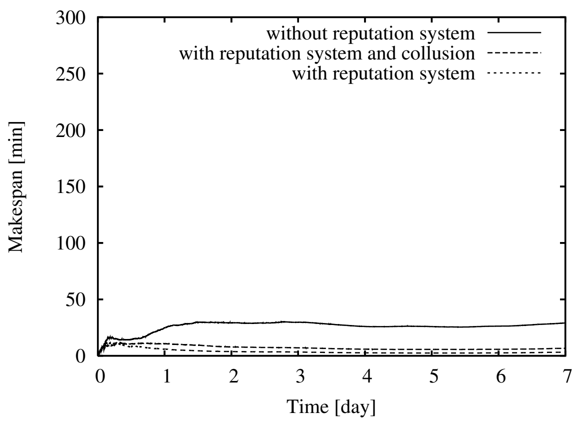

Fig. 3. Average makespan for jobs with 10 % free-riders in the Desktop Grid

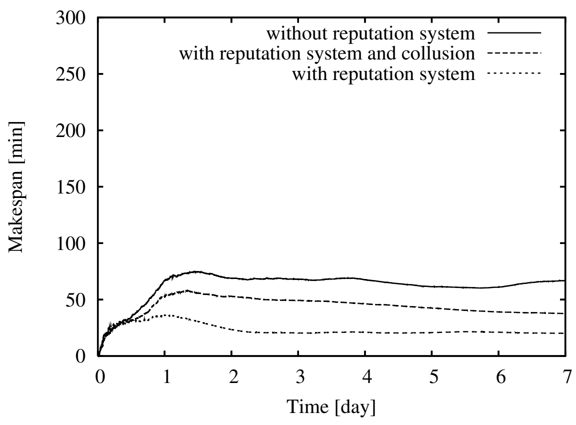

Fig. 4. Average makespan for jobs with 30 % free-riders in the Desktop Grid

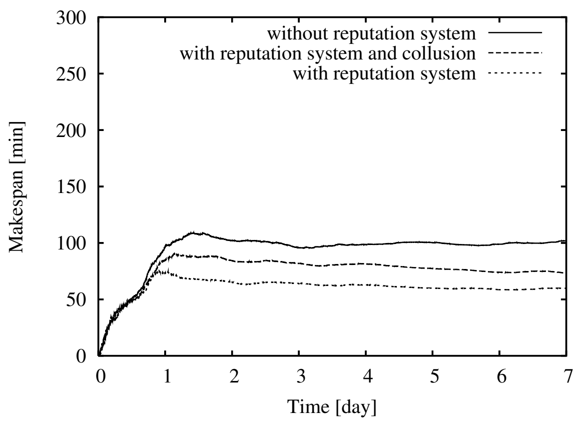

Fig. 5. Average makespan for jobs with 50 % free-riders in the Desktop Grid

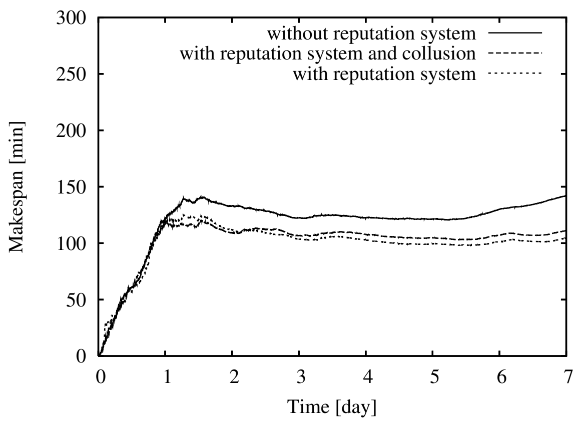

Fig. 6. Average makespan for jobs with 70 % free-riders in the Desktop Grid

makespan at a low level in the long term. This fact implies that our concept does actually exhibit the desired resilience towards free-riders.

From the figures, it can be seen that the reputation system resists a collusion but fares worse than with independent free-riders. Colluders, like regular free-riders, whitewash and return with a new identity 3 seconds after having left the Desktop Grid. If colluders take more time to return, the situation improves. Figure 7 depicts the average makespan for a free-rider fraction of 50 % when colluding peers take time to return to the overlay with a whitewashed identity after having left with their job completed. Their respective return time is randomly drawn from a uniformly distributed random variable bounded by the time limits given in the figure. While collusion still retains an impact on the average makespan, it is far less severe than in the aggressive case represented by a constant 3-second delay.

The handling of requests for participation in a scenario with 10 % free-riders is illustrated by Figure 8. It shows the cumulative numbers of tasks accepted and declined by worker peers, differentiated between free-riders and collaborators, the latter being either unprotected or protected by the proposed reputation system. The figure proves that in the unprotected case, collaborators' resources are quickly exhausted by the requests of free-riders, hence few resources remain for computing tasks initiated by other collaborators. In contrast, collaborators protected by the reputation system decline the vast majority of free-riders' requests. As desired, this behavior enables them to dedicate their resources to other collaborators.

The *speedup* of jobs has been depicted in Figure 9, with speedup defined as the ratio of the single-machine makespan to the makespan attained in the distributed computation case. This experiment has been conducted to quantify the general benefit of Desktop Grids which is to accelerate the computation of arbitrary jobs using otherwise idle resources. As also seen in Figure 3, the speedup of jobs drops to a low level already with as little as 10 % of free-riding peers if protection against free-riders is unavailable. The experimental results confirm that in the optimal case with all peers being altruists and zero overhead from a reputation system, a 100-peer Desktop Grid can generate a peak average speedup of approximately 60. The overhead generated by the reputation system exerts only a small influence on the speedup. This can also be seen in Figure 10 which relates to the average makespan: ensuring scalability, the proposed reputation system adds little overhead to the Desktop Grid's operations. Figure 10 depicts a comparison between two Desktop Grids

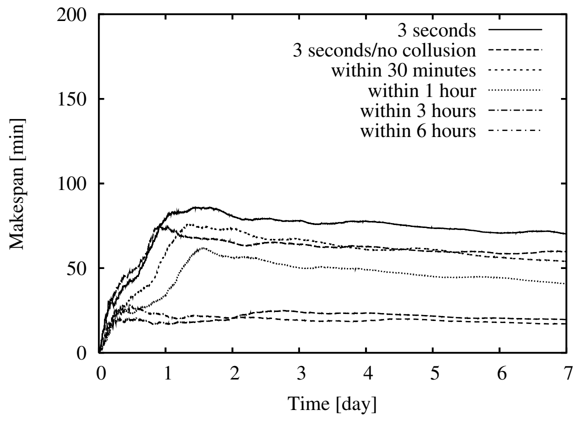

Fig. 7. Average makespan under collusion when colluders take time to return

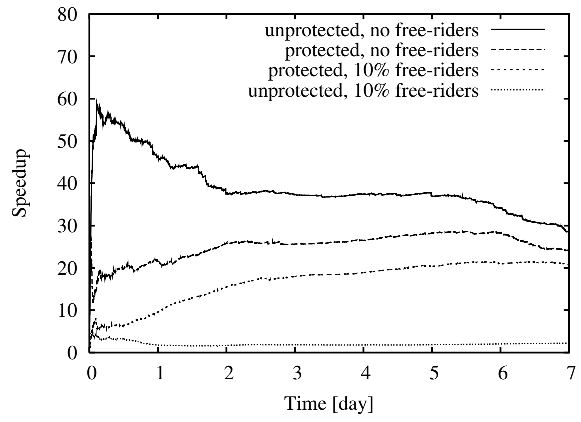

Fig. 9. Average speedup through distributed computation

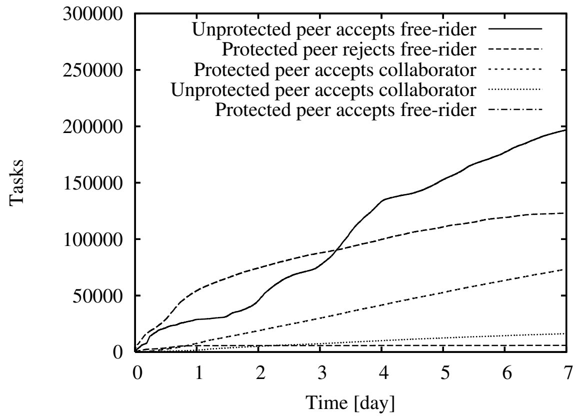

Fig. 8. Cumulative numbers of accepted and declined tasks

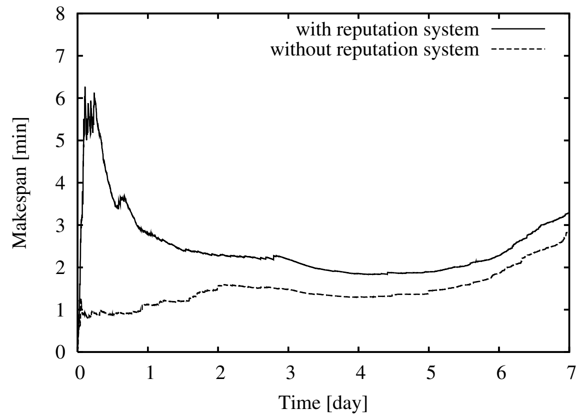

Fig. 10. Influence of reputation system on average makespan

populated with cooperative peers only: one completely unprotected system with all peers as altruists, and one system protected with our approach with all peers as collaborators. Since job volumes differ considerably, large-volume jobs can increase the makespan as seen in the figure. A volume-adjusted assessment of the overhead impact is given in Figure 11 which shows the average number of workers per job. Apart from the initial phase where collaborative peers first need to gain trust in each other under the control of the proposed reputation system, the difference in performance remains minor.

In summary, the simulation experiments confirm the effectiveness of the proposed reputation system. With only a minor effect on efficiency caused by the administrative overhead, the system enables cooperative peers to detect and avoid free-riders, fostering reciprocative behavior in the Desktop Grid.

VI. Related work

The benefits of self-organizing P2P structures to Desktop Grids have been pointed out in [6]. Decentralized control well suits Desktop Grids which can encompass millions of desktop computers. The approach in [6] is based on mobile agents forming a tree-like overlay. It considers altruistic behavior only, and its capability to deal with peer failures is limited.

A two-layered middleware for distributed computing in P2P overlays has been introduced in [28] by the name of *Vishwa.* Vishwa incorporates load migration, fault-tolerance, and a notion of proximity awareness as nearby peers form clusters, but it is still based on altruistic peer behavior and does not tap the benefits of super-peer overlays.

The free-riding phenomenon affects distributed systems whose benefit relies on voluntary contributions from its selfish participants. Free-riding has been observed in P2P file-sharing networks as a cause for performance degradation [16], [19], [21]. Due to a different nature of resources in Desktop Grids, solutions for free-riding prevention in file-sharing networks need to be modified for a Desktop Grid setting. While files may be distributed and stored, CPU cycles are volatile and can neither be stored nor replicated.

An application-neutral reputation scheme is introduced in [17]. It is based on solicited first-hand feedback acquired through time-to-live-bounded random walk sampling. Its utility for Desktop Grids is limited because of the expensive polling scheme and the inability to deal with considerable churn. As [19] confirms, a shared interaction history fits large-scale, churn-prone settings better than a witness-only approach.

The EigenTrust algorithm creates a global trust estimate in a distributed way, based on the concept of transitive trust [21], [40]. However, EigenTrust relies on a set of pre-trusted

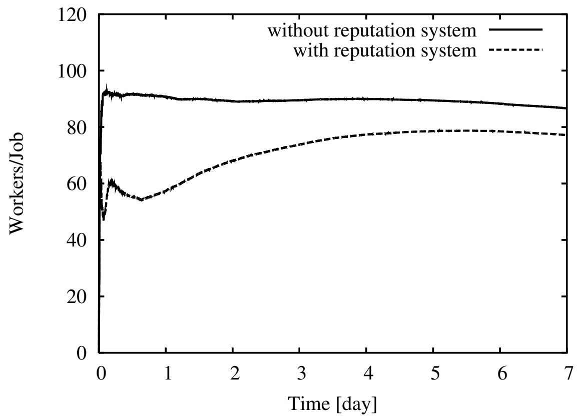

Fig. 11. Influence of reputation system on average number of workers per job

peers, does not incorporate the advantages of super-peers, and is susceptible to collusion.

OurGrid is a Desktop Grid which also incorporates a reputation mechanism, the *Network of Favors* [5], [18]. OurGrid is not super-peer-based, and the accumulated reputation does not decay over time. As with [17], reputation concerns direct witnesses of interactions in OurGrid only which is incompatible with substantial dynamics.

The framework introduced in [41] explicitly benefits from the presence of super-peers. Resorting to threshold cryptography, it focuses on the security aspects of feedback submission among peers. It does not intend to specify a reputation scheme itself, and due to its application-neutral scope, provides no insight into Desktop Grid workflows.

The reputation scheme proposed in [42] incorporates the notion of incentive-compatibility from the field of mechanism design. It solves the problem of forged feedback on other peers' past performances by seeing to it that telling the truth is in the best self interest of the reporting peers. The basic idea is to spread reputation information via specialized intermediary peers called R-agents. Instead of spreading own feedback directly to other peers, a peer buys reputation information from and sells its own experience with other peers to trustworthy R-agents. While it solves the problem of truthfulness, the scheme exhibits several drawbacks. It requires at least one R-agent to be present in the system at all times, it depends on all R-agents being trustworthy at all times, and it is susceptible to collusion. Also, the scheme suffers from the disadvantage of establishing truthfulness using payments according to the mechanism design approach: there needs to be a virtual currency which all peers in the system, including the R-agents, accept.

With respect to reputation systems, [29] distinguishes symmetric from asymmetric approaches. In symmetric systems, the computation of a remote peer's reputation happens under anonymity; peer identities can be exchanged as long as the trust graph's topology remains unmodified. Except for the trivial constant variant, symmetric reputation functions are vulnerable to collusion. In contrast, asymmetric reputation functions apply when every peer computes remote peer reputations by itself, and can be shown to withstand Sybil attacks under certain conditions. The reputation system proposed in this article uses an asymmetric approach.

A different form of free-riding in P2P networks concerns message routing in the overlay. If peers are unwilling to forward or answer queries, the overlay itself may not work as desired. Exploiting properties of the CAN overlay topology, [32] introduces a distributed reputation system designed to thwart uncooperative behavior in the context of message propagation.

In Desktop Grids, workers may cheat by choosing not to compute their assigned task but to return arbitrary data. To counter this, a replication approach might assign the same task to multiple workers and pick the correct result through majority voting. An alternative to this is Quiz [43] which assigns a number of tasks to one worker, including a Quiz task of which the submitting peer already knows the result. If the Quiz task is properly processed, the worker is assumed to have processed the other tasks properly, too. This sampling method is found to be superior to task replication. It is used in a Desktop Grid scheme which also includes reputation, *Cluster Computing on the Fly* [43].

VII. Conclusion and future work

Desktop Grids offer attractive opportunities for large-scale distributed computation. However, free-riding behavior may severely degrade a Desktop Grid's performance. In this article, we have presented a distributed reputation system for dynamic super-peer-based Desktop Grids with selfish peers. All components of our contribution – the reputation system, the underlying super-peer topology and the accompanying workflow model – are designed for scalability, and there is neither a single point of failure nor a need for pre-trusted nodes. Moreover, our concept effectively detects free-riding peers, encouraging cooperative peers to contribute resources. In simulations, we have empirically verified our proposed scheme's effectiveness. Future work includes the integration of a super-peer structure which interconnects the super-peers with a Chord ring [44], [45]. Moreover, to tackle the effects of collusion in scenarios with large fractions of colluding peers, we consider a multi-level similarity measure that incorporates the confidence into one's own ratings to better deal with collusion and less severe forms of spoof feedback. We also plan to use network coordinates for delay-optimized worker selection in CP scenarios where jobs benefit from inter-worker communication. Ultimately, we intend to deploy our proposed scheme on PlanetLab.

References

[1] Peter Merz, Florian Kolter, and Matthias Priebe. Free-Riding Prevention in Super-Peer Desktop Grids. In *Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology, ICCGI 2008, IARIA*, pages 297–302, 2008.

[2] Ian T. Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, pages 118–128, 2003.

[3] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[4] David P. Anderson and Gilles Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid*, pages 73–80, 2006.

[5] Nazareno Andrade, Francisco Vilar Brasileiro, Walfredo Cirne, and Miranda Mowbray. Automatic grid assembly by promoting collaboration in Peer-to-Peer grids. *Journal of Parallel and Distributed Computing*, 67(8):957–966, 2007.

[6] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. The organic grid: self-organizing computation on a Peer-to-Peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.

[7] Noriyuki Fujimoto and Kenichi Hagihara. A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks. In *Proceedings of the 2004 Symposium on Applications and the Internet Workshops*, pages 674–680, 2004.

[8] Derrick Kondo, Michela Taufer, Charles L. Brooks, Henri Casanova, and Andrew A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.

[9] Adriana Iamnitchi and Ian T. Foster. A Peer-to-Peer Approach to Resource Location in Grid Environments. In Jarek Nabrzynski, Jennifer M. Schopf, and Jan Weglarz, editors, *Grid resource management: state of the art and future trends*, pages 413–429. Kluwer, 2004.

[10] Beverly Yang and Hector Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the 19th International Conference on Data Engineering*, pages 49–62, 2003.

[11] Gian Paolo Jesi, Alberto Montresor, and Özalp Babaoglu. Proximity-Aware Superpeer Overlay Topologies. In Alexander Keller and Jean-Philippe Martin-Flatin, editors, *Proceedings of SelfMan'06*, volume 3996 of *Lecture Notes in Computer Science*, pages 43–57. Springer, 2006.

[12] Peter Merz, Matthias Priebe, and Steffen Wolf. A Simulation Framework for Distributed Super-Peer Topology Construction Using Network Coordinates. In *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 491–498, 2008.

[13] Peter Merz, Matthias Priebe, and Steffen Wolf. Super-Peer Selection in Peer-to-Peer Networks using Network Coordinates. In *Proceedings of the 3rd International Conference on Internet and Web Applications and Services*, pages 385–390, 2008.

[14] Russ Cox, Frank Dabek, M. Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. *Computer Communication Review*, 34(1):113–118, 2004.

[15] Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of IEEE INFOCOM*, 2002.

[16] Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[17] Emmanuelle Anceaume and Aina Ravoaja. Incentive-Based Robust Reputation Mechanism for P2P Services. In *Proceedings of the 10th International Conference on Principles of Distributed Systems*, pages 305–319, 2006.

[18] Nazareno Andrade, Francisco Vilar Brasileiro, Walfredo Cirne, and Miranda Mowbray. Discouraging Free Riding in a Peer-to-Peer CPU-Sharing Grid. In *Proceedings of the 13th International Symposium on High-Performance Distributed Computing*, pages 129–137, 2004.

[19] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for Peer-to-Peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 102–111, 2004.

[20] Michal Feldman, Christos H. Papadimitriou, John Chuang, and Ion Stoica. Free-riding and whitewashing in Peer-to-Peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1010–1019, 2006.

[21] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. Incentives for Combatting Freeriding on P2P Networks. In *Proceedings of the 9th International Euro-Par Conference*, pages 1273–1279, 2003.

[22] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, 06 2001.

[23] John Chuang. Designing incentive mechanisms for peer-to-peer systems. In *Proceedings of the 1st IEEE International Workshop on Grid Economics and Business Models*, pages 67–81, 2004.

[24] Michal Feldman and John Chuang. Overcoming free-riding behavior in Peer-to-Peer systems. *ACM SIGecom Exchanges*, 5(4):41–50, 2005.

[25] Philippe Golle, Kevin Leyton-Brown, and Ilya Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 264–267, 2001.

[26] Seth James Nielson, Scott Crosby, and Dan S. Wallach. A Taxonomy of Rational Attacks. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, pages 36–46, 2005.

[27] Robert Axelrod. *The evolution of cooperation*. Basic Books, New York, 1984.

[28] M. Venkateswara Reddy, A. Vijay Srinivas, Tarun Gopinath, and D. Janakiram. Vishwa: A reconfigurable P2P middleware for Grid Computations. In *Proceedings of the International Conference on Parallel Processing*, pages 381–390, 2006.

[29] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of Peer-to-Peer systems*, pages 128–132, 2005.

[30] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[31] Sebastian Kaune, Konstantin Pussep, Gareth Tyson, Andreas Mauthe, and Ralf Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, pages 10–22, 2008.

[32] Klemens Böhm and Erik Buchmann. Free riding-aware forwarding in Content-Addressable Networks. *The VLDB Journal*, 16(4):463–482, 2007.

[33] Jerry Banks, John S. Carson, and Barry L. Nelson. *Discrete-event system simulation*. Prentice Hall, Upper Saddle River, NJ, 1996.

[34] John R. Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer systems*, pages 251–260, 2002.

[35] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Technical report 2006-052, University of Massachusetts Amherst, Amherst, MA, 2006.

[36] Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on the World Wide Web*, pages 422–431, 2005.

[37] Audun Josang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[38] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias. The Interdomain Connectivity of PlanetLab Nodes. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement*, pages 73–82, 2004.

[39] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: an overlay testbed for broad-coverage services. *Computer Communication Review*, 33(3):3–12, 2003.

[40] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference*, pages 640–651, 2003.

[41] Tassos Dimitriou, Ghassan Karame, and Ioannis T. Christou. SuperTrust - A Secure and Efficient Framework for Handling Trust in Super Peer Networks. In *Proceedings of the 9th International Conference on Distributed Computing and Networking*, pages 350–362, 2008.

[42] Radu Jurca and Boi Faltings. An Incentive Compatible Reputation Mechanism. In *Proceedings of the IEEE Conference on E-Commerce*, pages 285–292, 2003.

[43] Shanyu Zhao, Virginia Lo, and Chris GauthierDickey. Result Verification and Trust-Based Scheduling in Peer-to-Peer Grids. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*, pages 31–38, 2005.

[44] Ion Stoica, Robert Morris, David Karger, Frans M. Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[45] Peter Merz, Steffen Wolf, Dennis Schwerdel, and Matthias Priebe. A Self-Organizing Super-Peer Overlay with a Chord Core for Desktop Grids. In K.A. Hummel and J.P.G. Sterbenz, editors, *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, volume 5343 of *Lecture Notes in Computer Science*, pages 23–34. Springer, 2008.