A Dynamic Approach for Characterizing Collusion in Desktop Grids

Louis-Claude Canon*§, Emmanuel Jeannot†§, Jon Weissman‡
*Nancy University, LORIA and LaBRI
Nancy & Bordeaux, France
Email: louis-claude.canon@labri.fr
†INRIA, LORIA, LaBRI
Nancy & Bordeaux, France
Email: emmanuel.jeannot@labri.fr
‡Dept. of Computer Science and Engineering
University of Minnesota, Twin Cities
Minneapolis, USA
Email: jon@cs.umn.edu

***Abstract*—By exploiting idle time on volunteer machines, desktop grids provide a way to execute large sets of tasks with negligible maintenance and low cost. Although desktop grids are attractive for cost-conscious projects, relying on external resources may compromise the correctness of application execution due to the well-known unreliability of nodes. In this paper, we consider the most challenging threat model: organized groups of cheaters that may collude to produce incorrect results. We propose two on-line algorithms for detecting collusion and characterizing the participant behaviors. Using several real-life traces, we show that our approach is accurate and efficient in identifying collusion and in estimating group behavior.**

***Keywords*-Desktop Grid; Collusion; Modeling; Sabotage**

I. Introduction

Desktop grids have proven to be a useful platform for many long-running scientific and engineering applications [1]–[4]. The attractive features of this paradigm include nearly infinite scale-out, low cost of deployment and management, and simplicity. However, these benefits do not come for free. The wide-dispersion, autonomous, and uncontrolled nature of the system, make this platform inherently unpredictable and unstable. Network links may go up and down, nodes may leave and join unpredictably including node failure, and nodes may fail to deliver correct and timely results. Much research has focused on how to mask this uncertainty under a set of assumptions, the most common is that failures are uncorrelated. With this assumption, tasks may be replicated across nodes to improve their reliability both in terms of timeliness (job completion time) and correctness (accuracy). General solutions for correctness assume a result cannot be certified in isolation and requires indirect validation through the use of pre-computed quizzes [5] or voting coupled with reputation systems [6]. The latter can be effective if the answer space is extremely large making the probability that two uncorrelated errors produce the same wrong result neglibly small. However, Internet-scale systems are rich with examples of correlated misbehavior and errors including botnet attacks, viruses, worms, sybil attacks, and buggy software distributions, to name a few. We believe that the long-running nature of desktop grid systems make them particularly vulnerable to such phenomena.

In this paper, we explore efficient and accurate techniques for representing, detecting, and characterizing the presence of correlated or collusive behavior in desktop grid systems. We distinguish between collusive behavior (correlated bad behavior) and agreement (general agreeement between nodes whether good or bad). Both concepts are useful for schedulers that wish to thwart collusion in the system. The principal challenge is to derive trustworthy global knowledge from unreliable sources. We present efficient data-structures and procedures that store and update collusion and agreement probabilities across arbitrary node groups. We assume a very aggressive threat model and our method requires only simple observations of worker behavior to be effective. We assess the accuracy of our approach using traces from well-known desktop grid applications. The results reveal that our approach is both accurate (the estimated collusion and agreement probabilities are close to the actual values) and efficient (the time required to compute these values across different nodes is small). In addition, colluding and non-colluding groups are accurately identified.

II. Related work

The goal of reputation systems is to associate reputation indices to each known worker. The literature in

§This work was done while L.C. Canon and E. Jeannot moved from the LORIA lab. (Nancy) to the LaBRI lab. (Bordeaux).

this area is rich and one of the best illustrative examples is the EigenTrust algorithm [7] which can be applied in desktop grids. Such methods do not target collusion detection and hence are not robust to an orchestrated attack.

Collusion estimation is also closely related to fault diagnosis [8] whose goal is to find faulty processors by performing a series of requests between pairs of processors. This problem is based on the assumption that processors are not hostile and do not try to deceive the detection mechanism. In this paper, we propose mechanisms based on observations that tolerate arbitrary erroneous behaviors.

Considering collusion with scheduling has been addressed by several projects. Zhao, Lo, and Gauthier-Dickey (see [5]) have analyzed two solutions based on redundancy and spot-checking and provide probabilistic analysis for both approaches. With their spot-checking mechanism, quiz tasks with verifiable results are inserted in the workload and allow the detection of cheaters. Ensuring that quiz tasks are indistinguishable from regular tasks, however, is a difficult problem. Similarly, Yurkewych, Levine, and Rosenberg propose a way to estimate the cost of a redundancy mechanism in the presence of colluders using game theory (see [9]). In this work, the goal is to determine the probability of auditing a task that minimizes the risk of collusion. These two works are based on the possibility of checking the result of a job. We do not consider this option as such a check is not always possible. However, such approaches are complementary to ours.

Silaghi *et al.* in [10] have proposed a technique that can thwart collusive behavior. The mechanism is based on redundancy and a reputation system that uses the EigenTrust algorithm. Workers detected as colluders are blacklisted and their previously computed jobs are resubmitted. However, this work has several drawbacks: it assumes that the detection algorithm is not known by the worker and it has to wait until the completion of all jobs before certifying the result. In contrast to this work, we propose a public algorithm where collusion characterization is improved each time a result is returned.

Therefore, to the best of our knowledge, this work is the first to tackle the problem of collusion characterization in the context of desktop grids using both general and practical assumptions.

III. Models and Definitions

A. Platform and Threat Model

We propose the following model of a desktop grid (see Figure 1), directly inspired from BOINC [1]:

- We are given a batch of *jobs* to be executed on the platform.

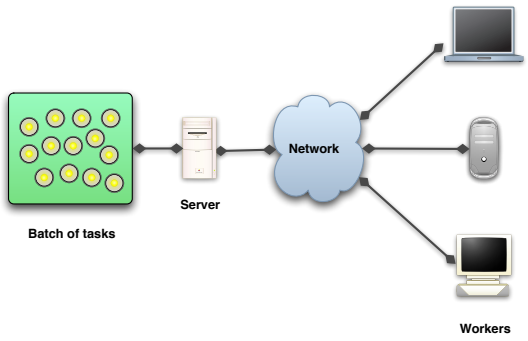

Figure 1: A Desktop Grid.

- We have n *workers*. Each worker $w \in W$ is able to compute all jobs in the batch. However, as workers may come and go they are not always available. This availability is defined by a set of intervals. If a worker, computing a job, leaves the system, it resumes job execution when it comes back.
- The *server* assigns each job to a set of workers and gathers the job results. For each job, we assume that there is only one correct result. Moreover, the result space is sufficiently large such that if two workers return the same incorrect result this means that they have *colluded*.

Collusion is defined as the cooperation of multiple workers in order to send the same incorrect result. We distinguish between two types of collusion. The first is when the saboteurs voluntarily cooperate to send more than one incorrect result trying to defeat the quorum algorithm. The other case is where workers do not deliberately collude as in the case of a virus or a bug in the code executed by the workers. Moreover it is possible that a worker (a colluder or not) may simply *fail* to correctly execute the job. To model these possibilities, we consider three worker behaviors:

- a worker may *fail* independently of others and return an incorrect result. The failure probability is fixed.
- a worker may belong to the *non-colluding group*. Such a *non colluding worker* never colludes with another worker but may fail. We assume that at least half of the workers are of this class.
- a worker may belong to one or more *colluding groups*. In order to reduce the chance of being detected, members of a group will act sometimes as colluders (return the same incorrect result) and sometimes as non-colluding workers (returning the same correct result). The probability that a group decides to collude or not is fixed. Moreover, it is possible that two different groups of colluders collude together, (*i.e.*, two workers from two different

groups may send the same incorrect result with a given probability). A worker in a colluding group may also fail independently (failures are predominant over collusion), and it returns an incorrect result alone.

We want to emphasize that the threat model proposed here is very strong: a worker may belong to one or more groups, groups can cooperate, colluders may sometime send a correct result to stay undetected, colluders are not required to synchronize to send the same incorrect result, and none of this information is known a-priori by the server.

B. Problem Definition

The problem we tackle focuses on characterizing the collusion properties of the workers, *i.e.*, for each worker we estimate the group(s) to which it belongs and collusion probability of each group and between groups. We make no assumptions about how jobs are mapped to workers other than jobs will be replicated. Although the way jobs are scheduled to workers can certainly help to detect and characterize the worker behaviors, here we make almost no assumption on how jobs are mapped to workers. Indeed, we want to propose a general strategy to solve our problem that must work for any (reasonable) scheduling algorithm. The only explicit assumption we make is that jobs are replicated for the sake of fault tolerance as in BOINC. Hence, the input of our problem is composed of two types of events:

- $< t, w, j, r >$ at time t, worker w returns result r for job j (*job result*).
- $< t, j >$ at time t, all the workers assigned to job j have finished their computation (*job completion*).

We assume that these events arrive in an order dictated by the worker speed, and thus, are time-stamped.

C. Group Model and Metrics

To account for the property that a worker may belong to several groups, we transform the input worker groups to a new set of groups. In the new groups, each worker belongs exactly to exactly one group and we update the collusion probability accordingly. For example, if workers w_1 and w_2 belong to group g and g' respectively, we put them in a third group g'' where the probability of collusion between g and g'' is the internal collusion probability of g. Hence, in the remainder of this paper, we presume that all workers are in exactly one group.

To model reality, we will group workers that share a common behavior. The output of our collusion characterization system will be a set of groups of workers that have the same collusion characteristics (*i.e* when a job is assigned to a subset of workers within this group, they always send the same results, correct or incorrect). To measure the accuracy of our solution, we will compare the estimation with actual collusion probabilities and group composition.

We denote by G the real group set and $\hat{G}$ the estimated group set. We assume that both sets are complete (every worker is in exactly one group). Let $O_w \in \hat{G}$ be the set of workers in the same group as worker w. We want the *observed groups* $\bigcup_{w \in W} O_w$ to agree with the real group of colluding workers and the observed probability to be close to the actual value. Combining both of these qualitative and quantitative measures into one accuracy metric is a difficult challenge. To do this, we proceed as follows. Let $g \in G$ and $g' \in G$ represent two real groups of workers. Let $e_{g \cup g'}$ be the absolute difference between the estimated probability that all the workers from set $g \cup g'$ return the same incorrect result for the same job, and the actual probability. We then use the RMSD (Root Mean Square Deviation) for aggregating the errors performed for each estimation:

$$\frac{1}{|G|} \sqrt{\sum_{(g,g') \in G^2} e_{g \cup g'}^2} \tag{1}$$

Smaller RMSD signals a more accurate group-based estimation. How to compute $e_{g \cup g'}$ depends on the way we internally represent $\hat{G}$. We propose two representations of $\hat{G}$ shortly, and we show how to bound $e_{g \cup g'}$ in section IV-A4. We consider two different aspects of accuracy:

- **convergence time**: time needed in order to achieve a desirable accuracy (defined by a threshold on the RMSD)
- **stabilized accuracy**: the accuracy achieved after a large number of events (median RMSD during the last 100 events of the simulation)

IV. Characterizing Collusion

The characterization and representation of collusion has several elements. We first present a way to model and calculate interactions between worker groups, colluding or non-colluding. The goal is to efficiently identify and update worker groups. For example, if all workers within a group return the same result, we will use only one entry to update the interactions. We then present the grouping algorithms based on two different representations, collusion and agreement, that each have different useful properties for schedulers that wish to thwart collusion. The dynamic/on-line algorithm we are proposing provides new estimation each time an event arrives.

The algorithm groups workers sharing similar characteristics. Events are treated as observations about the interactions between the groups (non-colluding or colluding) of workers. Thus, in some cases, as we work

W	Set of workers
n	Number of workers ($n = \|W\|$)
G	Real set of worker groups
$\hat{G}$	Observed set of worker groups
C	Real collusion matrix (C is of size $\|G\|$)
$\hat{C}$	Observed collusion matrix ($\|\hat{C}\| = \|\hat{G}\|$)
$\hat{A}$	Observed agreement matrix ($\|\hat{A}\| = \|\hat{G}\|$)
c_{ij}	Probability of collusion between group i and j
c_{ii}	Probability that workers of group i collude when assigned to the same job
$\hat{c}_{ij}$	Observed probability of collusion between group i and j
$\hat{a}_{ij}$	Observed probability of agreement between group i and j
O_w	Set of workers that are put in the same group as worker w ($O_w \in \hat{G}$)
L	Subset of workers ($L \subseteq W$)
K_L	Set of groups covering every workers of L ($K_L = \bigcup_{w \in L} O_w$)
E_L	Event that all workers of L have colluded to return the same incorrect result for a given job
$W_{r,j}$	Set of workers that have returned result r for job j

Table I: List of symbols

at the group level we will receive redundant information from individual workers. In this case the algorithm will take care to update its internal representation only once. For instance, if all the workers belonging to the same group return the same result, we will use only one entry to update the interaction between this group and the other groups.

A. Interaction Model

Each interaction between or within groups are represented by a random variable having a beta distribution. Using a beta distribution comes from the *Bayesian inference theory* in which observations are used to update or infer the probability of an hypothesis [11]. In our case, it works as follows. A beta distribution has two parameters α and β. The average value of that distribution represents the estimated probability of an interaction and is equal to $\frac{\alpha}{\alpha+\beta}$. At the beginning of the process, we set $\alpha = 1$ and $\beta = 1$. When an event reinforces an interaction, we increment the α parameter of the corresponding distribution and we increment the β parameter in the opposite case. Moreover, the *confidence interval* of a given distribution represents the confidence that we have in the corresponding estimation. This confidence interval is a function of α and β (*i.e.*, the number of observations).

1) Collusion-based Representation: The first representation of the interactions is based on estimations of collusion probabilities between groups. A matrix $\hat{C}$ contains the beta distributions corresponding to the collusion probability estimation between observed groups. The estimated probability that workers of groups i and j return the same incorrect result for a job is the mean of this distribution which we denote by $\hat{c}_{ij}$. This representation has one drawback. When two workers return the same result we need another mechanism to determine if this result is correct or incorrect to decide if there is collusion or not. Since this mechanism has to be based on the previous estimates, there is a risk that the system might amplify estimation errors and fail to converge. We propose an *adaptation* technique in Section IV-B1 that allows us to remedy this problem.

Let L be any subset of workers ($L \subseteq W$) and K_L be the set of groups covering all of the workers of L, *i.e.*, $K_L = \bigcup_{w \in L} O_w$. Let E_L be the event that all workers of L have colluded to return the same incorrect result for a given job. The following two lemmas are used to produce estimations on the collusive behaviors between any subset of workers L.

Lemma 1.

$$0 \leq \Pr[E_L] \leq \min_{(i,j) \in K_L^2} (c_{ij})$$

Proof: These bounds are trivially obtained from the definitions: probabilities are positive; the probability that all workers in L collude cannot exceeds pairwise collusion probabilities. ■

Lemma 2. *Let $K_L = i \cup j \cup k$ such that $(i,j,k) \in G^3$. Then:*

$$\max\left(0, \frac{c_{ij} + c_{ik} + c_{jk} - 1}{2}\right) \leq \Pr[E_L]$$

$$\Pr[E_L] \leq \min(c_{ij}, c_{ik}, c_{jk})$$

Proof: By definition, groups i and j either collude together with k, or they collude together without k: $c_{ij} = \Pr[E_L] + \Pr[E_{i \cup j} \cap \overline{E_L}]$. Analogously, $c_{ik} = \Pr[\overline{E_L}] + \Pr[E_{i \cup k} \cap \overline{E_L}]$ and $c_{jk} = \Pr[\overline{E_L}] + \Pr[E_{j \cup k} \cap \overline{E_L}]$. Also, $\Pr[E_{i \cup j} \cap \overline{E_L}] + \Pr[E_{i \cup k} \cap \overline{E_L}] + \Pr[E_{j \cup k} \cap \overline{E_L}] + \Pr[E_L] \leq 1$ because the events are disjoint. Hence:

$$\frac{c_{ij} + c_{ik} + c_{jk} - 1}{2} \leq \Pr[E_L]$$

The upper bound is obtained by applying Lemma 1 and by observing that $\forall (i,j), c_{ij} \leq c_{ii}$ (groups never collude more with others than they do internally). ■

The worst case scenario that maximizes the bounds given in Lemma 2 is achieved when $c_{ij} = c_{ik} = c_{jk} = \frac{1}{3}$. In this case, the range defined by the bounds is $\frac{1}{3}$.

2) Agreement-based Representation: The second representation is based on absolute observations, *i.e.*, agreements between groups. A matrix $\hat{A}$ contains the beta distributions corresponding to agreement probability estimation between the observed groups. The probability that workers of group i return the same result (correct or incorrect) as the workers of group j

is the mean of the corresponding distribution which we denote by $\hat{a}_{ij}$. With this absolute representation, there is no need for an external mechanism to determine if the result is correct and hence the system will converge.

The following two lemmas provides bounds for estimating the probability that a set of workers L collude together to return the same incorrect result, *i.e.*, $\Pr[E_L]$.

Recall that the largest group is indexed to be 1 and is assumed to be the non-colluding group.

Lemma 3. *Let $K_L = i \cup j$ such that $(i,j) \in G^2$. Then:*

$$\max(0, a_{ij} - a_{1i}, a_{ij} - a_{1j}) \leq \Pr[E_L]$$

$$\Pr[E_L] \leq \min\left(a_{ij}, \frac{1 + a_{ij} - a_{1i} - a_{1j}}{2}\right)$$

Proof: We separate the sample space into the following disjoint events:

$\overline{E_i} \cap \overline{E_j}$	neither group i nor j collude
$\overline{E_i} \cap E_j$	group j collude but not group i
$E_i \cap \overline{E_j}$	group i collude but not group j
$\overline{E_L} \cap E_i \cap E_j$	group i and j collude but not together
E_L	group i and j collude together

By construction, $a_{1i} = \Pr[\overline{E_i} \cap \overline{E_j}] + \Pr[\overline{E_i} \cap E_j]$, $a_{1j} = \Pr[\overline{E_i} \cap \overline{E_j}] + \Pr[E_i \cap \overline{E_j}]$, and $a_{ij} = \Pr[\overline{E_i} \cap \overline{E_j}] + \Pr[E_L]$. Since, all the sample space is covered by these events, the sum of their probability is equals to 1. Thus:

$$\Pr[E_L] = \frac{1 + a_{ij} - a_{1i} - a_{1j}}{2} - \frac{\Pr[\overline{E_L} \cap E_i \cap E_j]}{2}$$

$\Pr[\overline{E_L} \cap E_i \cap E_j] \leq 1 - a_{1i}$ because group i colludes less often than it agrees with the largest group (assumed to be a non-colluding group). Analogously, $\Pr[\overline{E_L} \cap E_i \cap E_j] \leq 1 - a_{1j}$ and the lower bound can then be deduced. The superior bound is derived by noting that probabilities are positive and the collusion probability between groups i and j cannot exceed their agreement. ■

The largest range of the bounds given in Lemma 3 is $\frac{1}{3}$ when $a_{ij} = a_{1i} = a_{1j} = \frac{1}{3}$.

Lemma 4.

$$0 \leq \Pr[E_L] \leq \min_{(i,j) \in K_L^2}\left(a_{ij}, \frac{1 + a_{ij} - a_{1i} - a_{1j}}{2}\right)$$

Proof: It is a direct generalization of Lemma 3. ■

3) Relation between both Representations: We now present some mathematical relations between these two representations.

Theorem 1. *Let $C = c_{ij}$ and $A = a_{ij}$ represent the actual group collusion and agreement matrices respectively. The largest group is indexed to be 1 and is assumed to be the non-colluding group. Then, we can bound A from C and C from A with the following equations:*

$$a_{ij} \leq 1 + 2 \times c_{ij} - c_{ii} - c_{jj}$$

$$c_{ij} \leq \frac{1 + a_{ij} - a_{1i} - a_{1j}}{2}$$

Proof: Groups i and j are agreeing either if they do not collude or if they collude together. Hence:

$$a_{ij} = \Pr[E_{i \cup j} \cup \overline{E_i \cup E_j}]$$

$$a_{ij} = \Pr[E_{i \cup j}] + 1 - \Pr[E_i \cup E_j]$$

$$a_{ij} = c_{ij} + 1 - \Pr[E_i] - \Pr[E_j] + \Pr[E_i \cap E_j]$$

$$a_{ij} = c_{ij} + 1 - c_{ii} - c_{jj} + \Pr[E_{i \cup j}] + \Pr[\overline{E_{i \cup j}} \cap E_i \cap E_j]$$

As there is the case where groups i and j collude individually, we end up with the following bound:

$$a_{ij} \leq 2 \times c_{ij} + 1 - c_{ii} - c_{jj}$$

Bounding matrix C from matrix A is an application of Lemma 4:

$$c_{ij} \leq \frac{1 + a_{ij} - a_{1i} - a_{1j}}{2}$$

■

4) Evaluation of the errors in Equation 1: Based on the above result we can now expand equation (1). We use the upper bound of the estimated probability of collusion of workers in $g \cup g'$ to compute $e_{g \cup g'}$. Hence, we have: $e_{g \cup g'} = |c_{g,g'} - \min_{(i,j) \in K^2_{g \cup g'}}(\hat{c}_{ij})|$ when using the collusion representation and $e_{g \cup g'} = \left|c_{g,g'} - \min_{(i,j) \in K^2_{g \cup g'}}\left(\hat{a}_{ij}, \frac{1 + \hat{a}_{ij} - \hat{a}_{1i} - \hat{a}_{1j}}{2}\right)\right|$ when using the agreement representation.

B. On-line Algorithm

Above, we have described the data structures used in the algorithm. We now describe how groups are built and updated. This is done through the *merge* and *split* of worker groups. Initially, each worker is put in a singleton group. After some interactions, there may be enough observations to determine a similarity between two groups, which are then subsequently *merged* into a single group. Since the conditions for merging groups of workers is statistical, erroneous merges can happen. In this case, a worker should be separated from a set of workers when it is observed to behave differently. We call this operation a *split*. The objective is to create a correspondence between the observed groups of similar workers, and the actual non-colluding and colluding groups. The conditions for merging and splitting are different in both representations (collusion and agreement). Note that due to merge and split the size of the agreement and collusion matrices change with time.

Algorithm 1: Dynamic merge and split with collusion representation

```
foreach event e do
    if e =< t, w, j, r > then // Job result
1       foreach v ∈ W_{r',j}, r' ≠ r do // v found a result different than w
2           if |W_{r',j}| ≠ 1 then // The result cannot be a failure because several workers have
            computed it
3               if it corresponds to an observation between 2 worker sets that was not already considered for job j then
4                   if O_v = O_w then // v and w computed two different results but are observed to be
                    in the same group
5                       split w from O_w
6                   else // v and w are not observed to be in the same group
7                       decrease the collusion probability estimation between O_v and O_w
8   else // e =< t, j > Job completion
9       execute external mechanism for certifying the best result among those generated for job j
10      foreach result r computed for job j do
11          if |W_{r,j}| = 1 then
12              the result might be a failure thus the observation is discarded
13          else
14              foreach (v, w) ∈ W^2_{r,j} do
15                  if the interaction between O_v and O_w has not already been considered for job j then
16                      if r is the certified result then
17                          Decrease the collusion probability estimation between O_v and O_w
18                      else
19                          Increase the collusion probability estimation between O_v and O_w
20                      if O_v ≠ O_w and merge is possible then
21                          merge O_v and O_w
```

1) Collusion-based Grouping: The procedure used with the collusion representation is depicted in Algorithm 1. We call $W_{r,j}$ the set of workers which have all computed r for job j. Each event is examined in chronological order as would be done in an online scheduling framework. The observed groups with which w disagrees are then considered (line 1). If the interaction is relevant and new, either a split happens because each worker in the same group should agree (line 5) or the collusion estimation is decreased as both groups disagree (line 7). The second kind of event denotes the termination of a job which triggers the certification mechanism (line 9). The proposed algorithm does not depend on any specific certification mechanism and hence this mechanism is not described here. However, it is expected that such a mechanism is sufficiently accurate to give relevant answers and ensures convergence. Once a result has been certified and thus considered to be correct, all of the results are examined (line 10). If only one worker has computed a particular result, it is possible that the system faces a failure and hence no update of the collusion probability is performed (line 12). Otherwise, we consider all pairs of workers that have returned this result (line 14). If the interaction between the observed groups O_w and O_v has not been accounted for (line 15), we decrease or increase the estimation of the collusion probability between these two groups based on whether r is the certified result or not (lines 16-19). Last, if the two groups are different but share common characteristics, we merge them (lines 20-21). In all the cases (line 7, 17, 19), the update of the collusion probability is done using the method described in Section IV-A.

By assumption, the largest group must be the non-colluding group (the percentage of colluders is assumed to always be below 50%) and hence the probability of collusion within this group must be 0. However, when a merge or split happens, the largest group may change and the internal collusion probability may not necessarily be 0. It is the job of the *adaptation process* to update the observed collusion probabilities to make them coherent. Without loss of generality, let 1 be the index of the new largest group. The adaptation process modifies the matrix $\hat{C}$ in two steps: the collusion matrix is first converted into an agreement matrix, without changing the composition of the observed group; bounds are then used to regenerate a collusion matrix $\hat{C}'$. From Theorem 1, we have the following inequality for each element of the new matrix:

$$\hat{c'_{ij}} \leq \hat{c_{ij}} - \hat{c_{1i}} - \hat{c_{1j}} + \hat{c_{11}}$$

As this is the best bound we have, we use it for computing the adapted matrix.

These arithmetic operations are actually performed

on the beta distributions related to the collusion estimations. Since addition operations over random variables propagate errors, the resulting distributions have less precision. Adaptation should thus be avoided if possible.

The merging condition is critical for the success of the algorithm. The key idea is to favor early but possibly imprecise merges and late but precise merges. Two groups need to be merged if they collude together with the same probability as when they are operating alone. Let us first describe some notation for specifying the merging condition between observed groups i and j (line 20). Let c_{ii}, c_{ij} and c_{jj} be the collusion estimates. $N(c)$ is the number of observations that led to c and e_c is its error interval. Let $\hat{m}$ be the number of observed groups. Then, i and j are merged if: $|c_{ii}-c_{ij}| < \frac{e_{c_{ii}}+e_{c_{ij}}}{2} \wedge |c_{ii}-c_{jj}| < \frac{e_{c_{ii}}+e_{c_{jj}}}{2} \wedge |c_{ij}-c_{jj}| < \frac{e_{c_{ij}}+e_{c_{jj}}}{2}$ and if $\min(N(c_{ii}), N(c_{ij}), N(c_{jj})) > \frac{\gamma \times \max(|i \cup j|, \frac{n}{\hat{m}})}{(1-|c_{ii}-c_{ij}|)(1-|c_{ii}-c_{jj}|)(1-|c_{ij}-c_{jj}|)}$. Where

$$\gamma = \begin{cases} 2.5 & \text{if the largest observed group will change due to this merge} \\ 1 & \text{otherwise} \end{cases}$$

The first set of conditions check the consistency of the values. The second condition imposes a minimal number of observations that depends on four principles. The γ value, found empirically, reduces the likelihood of merges if the largest observed group will change due to the merge (as it would then induce an adaptation which introduces uncertainties in the estimations). $|i \cup j|$ allows us to increase the precision requirement when observed groups grow larger. The idea of early and imprecise merge is meaningless when large observed groups already exist (this is the meaning of $\frac{n}{\hat{m}}$). Finally, the denominator takes into account the numerical similarities between the collusion estimations.

The value of 2.5 for γ is empirical. It has to be larger than 2 but values greater than 3 hinder the convergence speed. The chosen value has been experimentally tested and leads to a good compromise between accuracy and convergence speed.

An additional splitting mechanism takes place when an update is performed on the collusion probabilities. If the collusion estimation of the largest observed group has to be increased, then the worker responsible for the increase is split from the observed group.

2) Agreement-based Grouping: The agreement representation requires a simpler algorithm than the collusion case as shown in Algorithm 2. First, the agreement and disagreement observations are easy to detect. Moreover, the second kind of event ($< t, j >$) is not needed and there is no need for a result certification mechanism.

Merge and split operations are simpler. Observed groups i and j are merged only if no disagreement between the groups have been observed (*i.e.*, $\hat{a}_{ij} = 1$) and if the number of observations is greater than $|i \cup j|$. Splits only happen if two workers from the same observed group disagree.

V. Empirical Validation

Both approaches are compared empirically using large-scale (on the order of 1 million) real-life inputs (*i.e.*, values of $< t, w, j, r >$ and $< t, j >$). Since no complete trace is available for our study as of this writing, we aggregate different traces of several desktop grid projects with a synthetic model of threat in order to obtain complete traces. Both heuristics are then run with all of the generated traces.

A. Input Description

The first main source of traces used is the *Failure Trace Archive* (FTA [12]) which provides worker availability traces of parallel and distributed systems. In particular, we use availability and performance information from the SETI@home project. Additional projects we used include: Overnet, a P2P network; and, Microsoft, an enterprise network. While these traces allow us to assign jobs to machines, the job durations need to be defined to time-stamp the inputs. Michela Taufer *et. al.* [13] modeled the in-progress delay, *i.e.*, the computation time required by jobs in several desktop grid projects. Additionally, she provided us a workload trace of the docking@home volunteer project, which we have used.

Jobs are assigned to workers using a quorum-based scheduling algorithm such as the one used in BOINC. Namely, each job is first assigned to k workers. Whenever a quorum of q is achieved for a given job (q workers agreeing on an identical result), no more workers are assigned to this job. Moreover, a job cannot be assigned to more than l workers, though a large l may be needed to detect collusion. The performance heterogeneity is determined by normalizing the job durations for each worker. For example, we consider that a machine twice as fast as the mean speed will take half as long for a given job. Whenever a worker becomes unavailable, its current computation is postponed until it rejoins the network. Finally, a timeout is associated to each job (14 days) and to each worker computation (4 days).

We overlay a threat model over the trace which allows workers to return erroneous results. Any worker returns a correct result by default, except if it fails or if it colludes. Failures are modeled as follows: a percentage of workers are completely reliable; unreliable workers return a reliable result with the same reliability probability. Colluding groups are based on the same principle: several "fractions" of colluders, one for each group, are

Algorithm 2: Dynamic merge and split with agreement representation

```
foreach event e do
    if e =< t, w, j, r > then // Job result
        if |W_{r,j}| ≠ 1 then // The result cannot be a failure because several workers have computed it
            foreach v ∈ W_{r,j} do // v and w found the same result
                if it corresponds to an observation between 2 worker sets that was not already considered for job j then
                    increase the agreement probability estimation between O_v and O_w
                    if O_v ≠ O_w and merge is possible then
                        merge O_v and O_w
        foreach v ∈ W_{r',j}, r' ≠ r do // v and w found a different result
            if |W_{r',j}| ≠ 1 then // The result cannot be a failure because several workers have
            computed it
                if it corresponds to an observation between 2 worker sets that was not already considered for job j then
                    if O_v = O_w then
                        split w and v from O_w
                    else
                        decrease the agreement probability estimation between O_v and O_w
```

specified; for each group, a probability of collusion is specified.

Table II depicts the empirical settings. For each parameter, every tested value is used (with all the other parameters set to their default value) for generating a trace. For the reliability and the collusion related parameters, 4 scenarios for reliability and 9 scenarios for collusion are generated. The additional *Pair* setting means that 40% of the workers belong to one of two colluding groups of equal size. In the first group, the probability to collude is 0.2 while in the second group workers always collude. In addition, we have used 5 parts of the huge SETI@home trace at different time periods to yield separate traces. This leads to 27 scenarios plus the default one. For each scenario, the seeds used for random generation (workload model and threat setting) take 20 distinct values. We then generate 560 traces on which both heuristics are run.

B. Results Analysis

For the first plot (Figure 2), we show a typical run that illustrates the accuracy evolution with time. After less than 10 hours, the RSMD drops for both approaches and stabilizes until the end of the trace. For this trace, the agreement-based algorithm eventually regroups workers into 3 sets (of sizes 79, 20 and 1) which matches the actual sets except for the singleton. Additionally, the RMSD is largely impacted by the agreement estimation between the two main sets which is 0.8 when it should be 0.5 (however, there is no error in the internal agreement probability of 1 in both cases). The collusion-based algorithm produces the same sets of size 79 and 1. However, the set of colluders, of size 20, is separated into 3 sets of sizes 10, 7 and 3. It's probability of collusion is estimated to be 0.28 when it is 0.5 in reality, and it is the main source of error that impacts the RMSD. In both cases, the estimated worker group structure is accraute as there are no false positives and negatives (workers wrongly considered as colluders or wrongly considered as non-colluders).

Although both heuristics have converged in Figure 2, the collusion is *underestimated* (and the agreement *overestimated*). This phenomenon is related to the following conditions that must all hold: a collusion probability lower than 1, a low quorum value, and a small fraction of colluders. With these settings, there are many cases where a worker assigned to a given job produces a wrong result due to collusion that is not matched by any of the results produced for this job. This happens if this worker is the only one of its colluding group assigned to this job and it is treated as a buggy worker. In this case, the wrong result is considered as being due to unreliability and not collusion by our approach (see line 12 of Algorithm 1). This inability to determine the cause of an incorrect orphan result implies that even if our techniques converge, it will produce less accurate estimations. A simple solution would be to increment the quorum (to reduce the risk of having orphan results) or to resubmit the job to a new worker in the same group of the orphan. However, these workarounds concern the scheduling level and are thus left to future work.

The next set of figures show results across the different traces. Generally, it is expected that our heuristics converge before the end of the traces (for instance, 3 days for most of the traces). We have empirically observed that when the RMSD is below 0.2, the structure of the worker groups is correctly estimated (*i.e.*, observed groups are close to the real groups).

Figures 3 to 8 illustrate the convergence time and stabilized accuracy measures for both heuristics. Each figure depicts the effect of the variation of one parameter, the others being set to their default value. The

Parameter	Default value	Tested values
Worker availability trace	Seti@home	Overnet, Microsoft, ...
Workload model or trace	charmm	mfold, docking@home
Quorum (k, q, l)	(4, 3, 10)	(2, 1, 2), (19, 15, 19)
Number of workers (n)	100	30, 50, 70, 80, 200
Reliability (fraction, probability)	(0.7, 0.7)	$\{0.7, 0.99\} \times \{0.7, 0.99\} \setminus (0.7, 0.7)$
Collusion (fraction, probability)	(0.2, 0.5)	$\{0.02, 0.2, 0.49\} \times \{0.01, 0.5, 1\} \setminus (0.2, 0.5)$, Pair

Table II: Experimental Paramaters

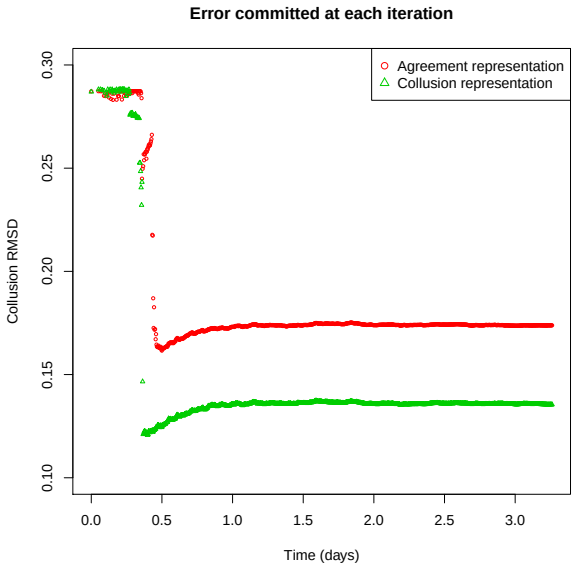

Figure 2: Analysis of a specific run with default settings.

results are graphically represented through the use of violin plots. These plots are a combination of box plot and kernel density plot. The box plot is the internal line and rectangle in the painted area. The rectangle extends from the median (the white mark) in both directions to the first and third quartile. The line extends to 1.5 times the interquartile range beyond each of the quartiles. Intuitively, most measures are concealed in the area where the line is drawn and 50% of them are in the rectangle. The kernel density is an estimation of the density of measured points. It can be thought as a smoothed histogram. The larger an area, the more points are contained within it. Violin plots are useful when multiple modalities exists, *i.e.*, when measures are regrouped in multiple locations.

Several traces are used and the efficiency of our methods is represented in Figure 3. We see that simulations behave similarly for each SETI@home trace. Namely, both heuristics converge in less than half a day in most of the cases. Additionally, the stabilized accuracy is better with collusion, though with agreement it is still within a tolerable range. With the microsoft trace, the convergence time is lower. Indeed, participants are more active in an entreprise network and thus there are more interactions. Although the overnet trace reveals similar stabilized accuracy than with the other traces, the convergence times are strangely distributed. This has yet to be investigated.

Several workloads are represented in Figure 4. We can draw the same conclusions about the stabilized accuracy, namely, collusion is better than the agreement representation and results are independant of the workload. The convergence times for the docking@home project are out of range because each job takes much more longer as compared to the charmm and mfold workload models. For these two last, however, the convergence time is very similar.

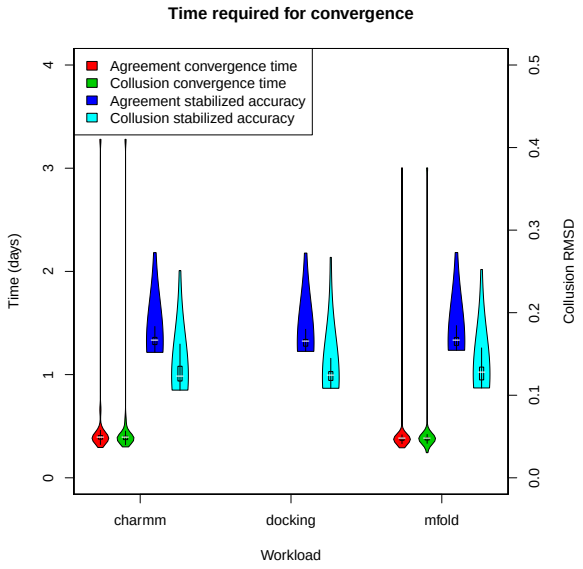

Figure 4: Workload model or trace

The effect of the quorum is represented in Figure 5. As expected, a quorum of one does not allow any of our approaches to converge. Indeed, with such a quorum, there is no interactions between worker groups (for building an observation, at least four results distributed in two sets of equal size are needed). Although the convergence time is similar with a quorum of 15, the

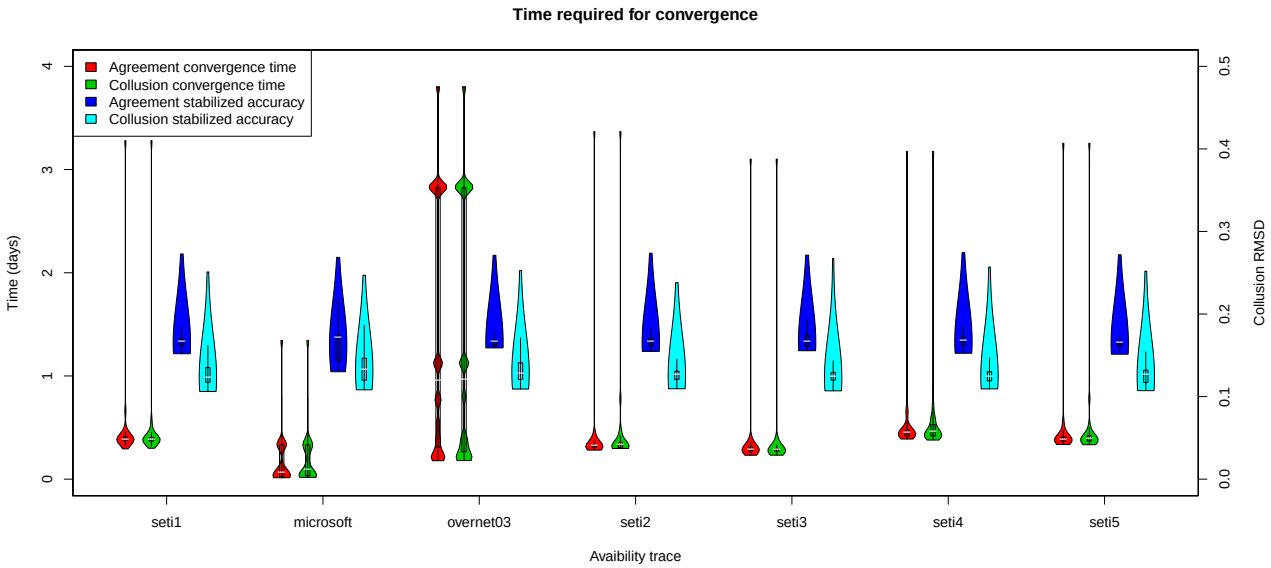

Figure 3: Availability traces

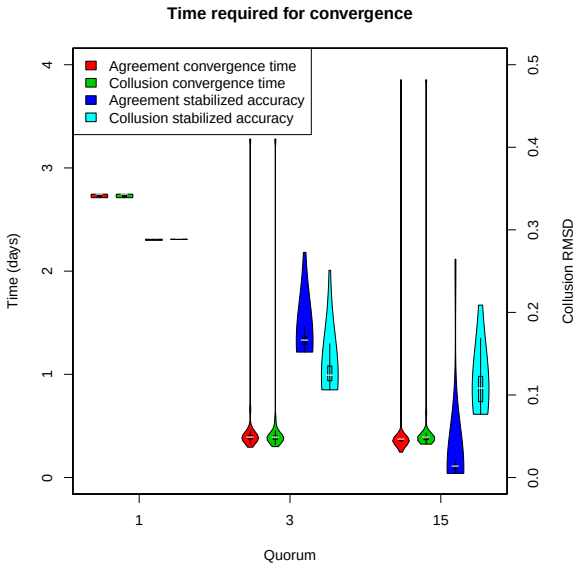

Figure 5: Quorum effect

agreement representation becomes extremely precise. Intuitively, this is due to the increase in observations that are done.

Figure 6 depicts the efficiency of our methods when the number of workers in the system increases. Results are stable until there are more than 100, with a slight increase in convergence time. With 200 workers, half of the runs do not converge before the end of the traces which explains why the precision degrades. Also, the maximum convergence time decreases. Indeed, with 1 million events in the input trace, time advances more slowly if there are more workers and it takes a longer time to get a accurate estimate of this environment.

In Figures 7 and 8, we vary both the fraction of colluders and their probability to collude. For example, in the first column on Figure 7, the collusion probability is 0.01, but the collusion fraction is either 0.02, 0.2, or 0.49. The scenario where the probability is 0.5 and the fraction is 0.02 appears to be the most difficult. Neither of our approaches are able to converge and the upper modality in both plots corresponds to this setting. In every case, specifying a probability of 0.01 leads to a good accuracy with fast convergence relative to the RMSD metric. However, for such small probability values, the RMSD is perhaps not the best way to measure accuracy. Indeed, our algorithms converge to estimations equal to 0 for which the RMSD (that computes absolute difference) is very low. Actually this setting is very hard to detect and the quick convergence indicates that our techniques do not detect the collusion. These low values explain the lower modality in Figure 8. On the other hand, colluding almost surely allows an easier and more precise detection for both heuristics than when it occurs with a 0.5 probability. The high precision indicates that the heuristics performs well when the probability value to estimate is close to 1. Another reason for which a probability of 1 is easier to detect is that it requires statistically fewer observations to be precisely estimated. It was expected that having a near majority of colluders (49%) would be harder to

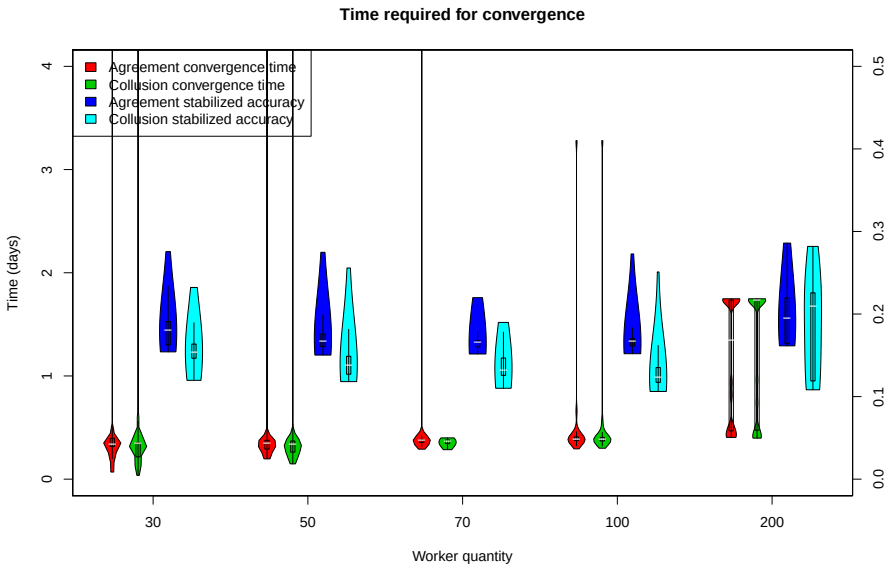

Figure 6: Worker quantity effect

detect than with a significant but limited fraction of colluders. Although there are indeed some cases with the 0.49 fraction that do not converge, the difference is small with the 0.2 fraction. Finally, dealing with a pair of colluding groups does not present a challenge for our approach.

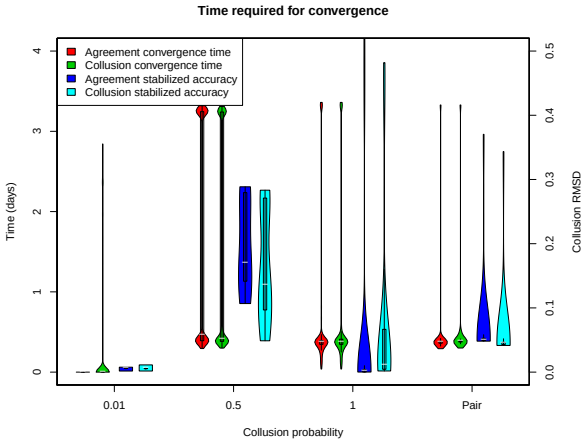

Figure 7: Collusion probabilities effect

Lastly, as our systems rely on the use of beta distributions, they are able to give an error interval for each estimation called the *introspection*. We assume that such an interval is an estimation of the error and we measure the accuracy of the introspection (the error on the error estimation). All these errors are aggregated with the RMSD (between the estimated error and the real error)

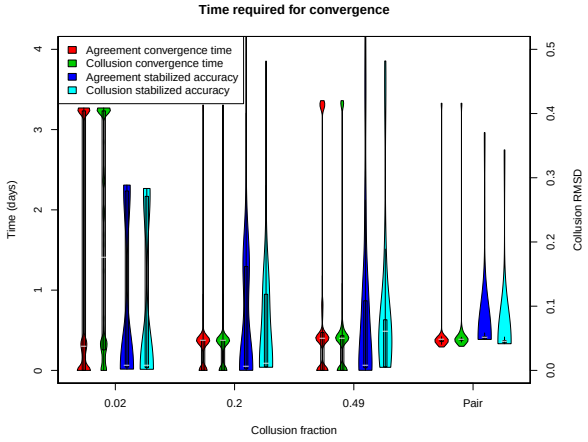

Figure 8: Colluder's fraction effect

and their statistical repartition for each run is shown on Figure 9 as an empirical cumulative distribution function (ECDF). Each value is the median RMSD over all iterations for one specific run. All of the values greater than 0.22 (8% of the measures) correspond to settings for which our algorithms do not converge. 63% of the error RMSD are between 0.1 and 0.22. This is related to the fact that, in most of the settings, the final collusion RMSD is subject to these bounds.

Finally, better introspection is observed for the collusion representation.

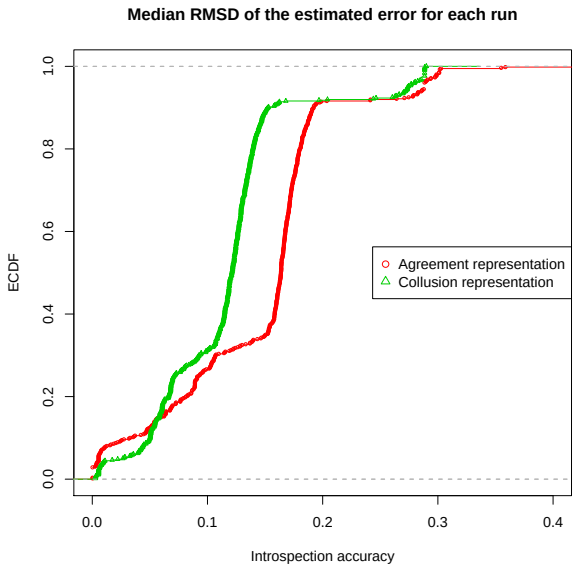

Figure 9: Empirical cumulative distribution function of the introspection accuracy for all runs.

VI. Conclusion

Coordinated attacks against a desktop grid remain a major threat to their correct functioning. In this paper, we proposed two algorithms for detecting and characterizing collusion in a desktop grid. Despite the fact that the threat model is very strong (a worker may belong to one or several groups, groups can cooperate, colluders may sometime send a correct result to stay undetected, colluders are not required to synchronize to send the same incorrect result, and no information is known a-priori by the server), the input events are minimal (we only know which workers have returned a given result for a given job) and that no assumption is made on the way jobs are scheduled to workers, the proposed solutions are very effective. Indeed, experimental results show that they are able to accurately and rapidly characterize collusion, most of the time. Among the two algorithms, we see that the collusion method is slightly better than the agreement method especially in terms of accuracy. However the agreement method is much simpler to implement and does not rely on any certification mechanism.

Future works are directed towards non-stationarity (when worker behavior changes with time). A simple method would be to reset the probabilities from time to time. Other techniques for non-stationarity including aging of prior observations to enable more rapid transitions will be studied. We also plan to work on the design of a scheduling method for a desktop grid that would defeat collusion. Indeed, we think that our characterization mechanism is accurate and fast enough to be used at the scheduling level. Finally, we anticipate that a scheduling algorithm could improve the collusion characterization by adapting the replication ratio according to its estimated accuracy.

References

[1] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *GRID*. IEEE Computer Society, 2004, pp. 4–10.

[2] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An Experiment in Public-Resource Computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[3] "Climateprediction.net," http://climateprediction.net/.

[4] "Folding@home," http://folding.stanford.edu/.

[5] S. Zhao, V. M. Lo, and C. Gauthier-Dickey, "Result verification and trust-based scheduling in peer-to-peer grids," in *Peer-to-Peer Computing*. IEEE Computer Society, 2005, pp. 31–38.

[6] J. D. Sonnek, A. Chandra, and J. B. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 11, pp. 1551–1564, 2007.

[7] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 640–651.

[8] E. P. D. Jr. and T. Nanya, "A hierarchical adaptive distributed system-level diagnosis algorithm," *IEEE Transactions on Computers*, vol. 47, no. 1, pp. 34–45, 1998.

[9] M. Yurkewych, B. N. Levine, and A. L. Rosenberg, "On the cost-ineffectiveness of redundancy in commercial p2p computing," in *ACM Conference on Computer and Communications Security*. ACM, 2005, pp. 280–288.

[10] G. Silaghi, P. Domingues, F. Araujo, L. M. Silva, and A. Arenas, "Defeating Colluding Nodes in Desktop Grid Computing Platforms," in *Parallel and Distributed Processing (IPDPS)*, Miami, Florida, USA, Apr. 2008, pp. 1–8.

[11] A. Jøsang, "The beta reputation system," in *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[12] "Failure trace archive," http://fta.inria.fr/.

[13] T. Estrada, M. Taufer, and K. Reed, "Modeling job lifespan delays in volunteer computing projects," in *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 331–338.