

On the Cost-Ineffectiveness of Redundancy in Commercial P2P Computing

Matthew Yurkewych Brian N. Levine Arnold L. Rosenberg
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
{yurk,brian,rsnrbg}@cs.umass.edu

ABSTRACT

We present a game-theoretic model of the interactions between server and clients in a constrained family of commercial P2P computations (where clients are financially compensated for work). We study the cost of implementing redundant task allocation (redundancy, for short) as a means of preventing cheating. Under the assumption that clients are motivated solely by the desire to maximize expected profit, we prove that, within this framework, redundancy is cost effective only when collusion among clients, including the Sybil attack, can be prevented. We show that in situations where this condition cannot be met, non-redundant task allocation is much less costly than redundancy.

Categories and Subject Descriptors

C.2.0 [General]: Security and protection; C.2.5 [Local and Wide-Area Networks]: Internet

General Terms

Security, Theory

Keywords

P2P Computing, Game Theory, Redundancy, Collusion

1. INTRODUCTION

The Internet has enabled a new paradigm of high-performance computing that costs a fraction of its alternatives [19] called *P2P computing* (P2PC) [15]. In these distributed systems, a server registers a set of clients across the Internet, allocates tasks to each client, and collects the results. P2PC with volunteer clients has many successful instantiations, including *fightAIDS@home* [13] and *folding@home* [14]. *Commercial* versions of P2PC, where clients are *paid* for computing tasks, has not yet achieved the same level of success.

However, efforts such as Google Compute [18] provide examples of projects that might one day prove the viability of commercial P2PC. In such systems, clients would receive some *payoff* for providing correct results.

Because P2PC systems work with untrusted clients, a fundamental problem is protection against *cheating* [1, 2, 16, 17]. Within this context, cheating means returning a result without completing the entire underlying computation [16, 17]. In order to ensure the correctness of the computation, the server must be able to identify cheating clients and their associated results.

In *commercial* P2PC, the overall expense of the endeavor [19] and the prevention of cheating are inseparable. That is, only affordable mechanisms for preventing cheating are viable. The most common mechanism for cheat prevention in P2PC systems is *redundant task allocation* [1] (*redundancy*, for short). In this approach, the P2PC project server issues each task to multiple clients, gathers the results, and obtains the answer it considers correct using some voting mechanism. Redundancy has been engineered into the open-source BOINC software [3], its popular instantiation SETI@home [2], and many other large P2PC projects [1].

What is troubling about this use of redundancy is that, on its own, this mechanism cannot guarantee correctness. Malicious clients can *collude* with one another—by agreeing to return the same answer (whether it has been correctly computed or not)—in order to increase their payoff without increasing their costs. On the Internet, multiple clients involved in a collusion scheme may in fact be a single devious entity who has generated numerous client identities and synchronized their outputs. When used to defeat redundancy, this strategy is an example of the *Sybil attack* and it may be practically impossible for the server to prevent without instituting a costly centralized authority to certify identities [9].

In this paper, we develop a game-theoretic model of P2P computing that includes different task allocation strategies, the costs of computing, and the possibility of the Sybil attack. Our main results prove that, within our environment, redundancy is a cost-effective mechanism for guaranteeing correctness only when collusion among clients can be prevented. If collusion cannot be prevented, which is likely to be the case [9], then non-redundant task allocation is much less costly than redundancy for protection against cheating clients. Our contributions are explained in more detail below.

Our contributions. Our analysis of P2PC is based on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS'05, November 7–11, 2005, Alexandria, Virginia, USA.
Copyright 2005 ACM 1-59593-226-7/05/0011 ...\$5.00.

our introduction of the *Internet Auditing Game*. We consider three task-allocation scenarios that a server might encounter in a real-world P2PC project.

1. Redundant allocation when collusion among clients is unrestricted.
2. Redundant allocation when collusion is prevented among clients.
3. Non-redundant allocation, i.e., each task is issued to a single client.

In order to render our framework mathematically tractable—so that we can *prove* the cost estimate we seek—we restrict the P2PC setting as follows. First, we assume that *cheating* by a client consists of fabricating the result of a task in order to deceive the server. Second, we focus on a class of computations where, for each task, there is a fixed bound on the probability that a cheater can survive an “audit” (i.e. a spot-check by the server) while cheating. This bound is known by both the server and the client. Third, we ignore the possibility of systematic errors that can plague P2PC in practice [1, 2, 3].

Our constraints preclude situations in which, unbeknownst to the server, a client has access to an algorithm that significantly improves her odds of cheating. However, the assumption that the probability of cheating is bounded is common to many mathematical and cryptographic P2PC projects. In fact, P2PC projects based on the RSA Key Search [10, 28] explicitly invite clients to devise such a cheating algorithm. (This could be interpreted as a measure of confidence in the unproven assumptions used in that particular method of cryptography.) See Section 3.2 for further discussion.

Under the above restrictions, we determine two costs that the server incurs: (1) the compensation that the server pays each client for its computing resources, i.e., the cost of *untrusted* computational resources; (2) the cost of resources the server uses to perform audits of returned results, i.e., the cost of *trusted* computational resources.

Under the idealized assumption that clients are *rational* [16, 17, 30]—hence seeking only to maximize their expected profit¹—we determine the minimum amount of auditing that the server must employ in order to guarantee that all clients will prefer an honest strategy over a cheating strategy. We thereby obtain an estimate of the minimum cost the server must allocate for trusted computational resources. We estimate the minimum cost that must be allocated to untrusted computational resources as the aggregate per-task reward, which is minimized when the poll size is minimized.

Given this environment, our main results prove that redundancy is a cost-effective mechanism for guaranteeing correctness only when collusion among clients can be prevented. Otherwise, non-redundant task allocation is much less costly than redundancy.

While our constraints and idealizations restrict the generality of conclusions we can draw from our study, the fact that our conclusions emerge from proofs rather than experimental analyses allow the reader to port our analyses to other scenarios.

¹The “expectation” governing profits is based on the probability that the server will audit any particular set of results; see Section 3.

2. RELATED WORK

Our work follows the recent trend of using game-theoretic mechanisms to model interactions between processors in distributed computations [12]. The technique we use is sometimes referred to as *Distributed Algorithmic Mechanism Design* (DAMD) [12]. The work that is most relevant to ours is an application of DAMD that focuses on algorithms for implementing redundancy in the presence of collusion, by Golle and Stubblebine [17]. Their work shows that under similar idealized conditions, probabilistic algorithms can be effective for redundant task allocation (with no auditing). Our work has also benefited from other studies that apply DAMD. Shneidman and Parks [31] examine redundancy in an application of DAMD for Interdomain Routing. A widely read application of DAMD that focuses on client-side methods for preventing cheating in P2PC is the study by Golle and Mironov [16].

As mentioned earlier, redundancy is gaining popularity in P2PC projects. An overview by Anderson [1] gives an account of this while naming popular projects that use redundancy. The open source BOINC software [3] described by Anderson [1] implements redundancy. So too does the popular P2PC project SETI@home [2], which is based on the BOINC software [1]. Redundancy is also used in Grid computing, which is a modality of distributed computing on the Internet that shares many security issues with P2PC. We refer to surveys by Foster et al. and Ledlie et al. for more on this connection [15, 22].

Our game-theoretic model is derived from the traditional tax-auditing game that is often used to motivate discussions of mixed-strategy Nash equilibrium. A basic description of this game is found in two common texts [26, 27], and further development is found in Mookherjee and Png [23]. For background on how a game-theoretic mechanism can reasonably model a realistic P2PC project, Shneidman and Parks [30] offer a connection between the “rational” players in DAMD and clients in real-world P2P networks. We feel that their study offers grounds for our idealized assumption that every client in the Internet Auditing Game behaves rationally.

The Sybil attack plagues many distributed systems that rely on untrusted clients. These include P2PC, anonymous communication systems [8], distributed file systems [7], reputation systems [5, 11], and routing protocols for ad hoc wireless networks [29] among others. Douecer provides the best discussion of the attack [9]. He shows that prevention of the attack is difficult without a centralized authority to certify identity, which is generally impractical for P2PC projects.

3. THE INTERNET AUDITING GAME

This section introduces the Internet Auditing Game that we use to model a commercial P2PC project. We begin with a detailed account of the origins and mathematical underpinnings of the Game.

3.1 The Traditional Auditing Game

In the game-theoretic literature [26, 27], one encounters the “tax auditing” game (possibly under a different name). Here, a tax authority plays against a citizen by offering a reward and penalty scheme for a completed tax return. The authority (who is also the auditor) may opt to audit the output of the citizen (the auditee). It is assumed that both

the auditor and auditee are *rational*, in that both are motivated solely by financial self-interest. It is also assumed that they are not *risk-seeking*, which means that they will choose an action only if the expected profit of that action is greater than the profit of all other actions. If the profit of some actions are equal, then the participant is indifferent between those actions.

In the *simultaneous-move* version of the game, the auditor has two options (jointly called the auditor’s *strategy set*): it can either *audit* or *trust* the auditee. Auditing is the more costly of these actions to the auditor. Similarly, the auditee has two options: it can either *cheat* or *be honest*. Honesty is the more costly action for the auditee.

Depending on the profit profile, the game will reach a particular mixed-strategy Nash equilibrium if the strategy sets are finite [24, 25]. Here, each player has a randomized equilibrium strategy that yields the best expected profit, given that the other player’s strategy is also a randomized equilibrium strategy. The game can be converted into a sequential move game where the auditor first announces the probability of auditing, and then the auditee responds. The probability announced by the auditor will indicate whether the auditor is truly indifferent among its equilibrium strategies. With sufficiently high probability, the auditor can influence the decision-making of the auditee to ensure that the auditee maximizes expected profit by always being honest [26, 27].

3.2 The Internet Auditing Game

We now adapt the traditional auditing game to obtain the Internet Auditing Game (*IAG*, for short), which pits a server against a set of clients. In the IAG, the server plays the role of the auditor in the traditional game, while the (colluding) clients play the role of the auditee. When there is no collusion among clients, the role of auditee is played by a single client.

The basic structure of the IAG. In order to perform (typically massive) computation, a *server* registers a set of *clients* across the Internet. It then allocates tasks from the computation to the clients, and it collects the results when they are returned by the clients. Clients are not rigorously authenticated, hence they are untrusted. They may behave in a variety of ways, including forming *teams* for the purpose of *colluding*. Indeed, at any moment, the state of the IAG is as follows. There are N clients, who are partitioned into k teams, T_1, \dots, T_k of respective sizes t_1, \dots, t_k . (Note that we assume that no client colludes with two teams on any play of the IAG.) Each team has a controlling force that dictates the single action taken by all its member clients. (As noted earlier, the “different” clients on a team may be distinct identities assumed by just one actual entity, say by spoofing IP addresses.) The server counters the possibility of such collusion via a “carrot” and a “stick.”

The server’s “carrot” consists of a fixed payment to a client for each task whose result is accepted by the server; the “stick” has two components. First, the server has the ability to allocate tasks redundantly, to multiple clients. The server can then compare the results returned for a given task, using some sort of voting scheme to choose the result it will consider correct. To enhance the effectiveness of such redundant allocation, the server, with the knowledge of the clients, has the option of employing some trusted computational resource to verify (audit) selected results from clients. We assume that the server has a fixed budget for computing a task.

From this budget, it allocates $Cost_{trusted}$ units toward auditing and $Cost_{untrusted}$ units toward client reimbursement. Of course, for P2PC to be worthwhile to the server, the per-task budget must not exceed $Cost_{trusted} + Cost_{untrusted}$; otherwise, the server might be better off using alternatives to a P2P framework.

Since the clients are aware that redundancy is being used, clients may form colluding teams in order to enhance their return for resources expended. In the IAG, this enhancement assumes one of two forms. Either the clients in a team have one client “X” actually perform the computation but have all team members return X’s result. Or, the clients in a team “cheat” by having one client “Y” fabricate a result and have all team members return Y’s fabrication.

- *The server’s goal is to guarantee that the computation gets done correctly and within its budget.*
- *The clients’ goal is to maximize their expected returns.* (In particular, they are not destructively malicious.)

The server achieves its goal by setting a suite of parameters and announcing them to the clients. Foremost among these parameters are the per-task, per-client reward and the probability of auditing a result. Impacting both of these parameters is the commonly known probability of being able to cheat on a task.

We now provide more detail on the IAG.

Players’ strategy sets. In the traditional auditing game, the auditee has the option of cheating or being honest. At an abstract level, these two strategies find counterparts in the IAG. As noted earlier, we identify cheating with fabricating an answer to fool the server, where the probability of success is known to be bounded. Honesty is straightforward: the client does the computation and submits the correct result. We do not consider any other strategies, although more information about the computation may admit additional cheating strategies. For illustration, Golle and Mironov [16] model a distributed computation after the RSA Secret Key Challenge [28].

Thus, the strategy set for a team T_j of clients consists of the following: T_j will either have one member fabricate the correct answer with the appropriate certain probability, or it will have a single member compute the correct answer; in both cases, the result becomes the unanimous answer for the team. The server S has the option of auditing a result—which it does with fixed probability α that is announced to the clients—or it trusts a result, with probability $1 - \alpha$.

The computation. To the end of mathematical tractability, we assume that the server knows enough about the computation to place a strict upper bound on the probability of cheating for each instance. In particular, this means that clients do not have access to an algorithm that allows them to improve their odds of cheating (above this bound) without the server’s knowledge. However, for many computations, such an algorithm either does not exist or has consequences well beyond the P2PC project if it does exist. For instance, the probability of cheating is *known* to be bounded for some important cryptographic functions [4, 21]. Additionally, if algorithms for cheating in projects based on the RSA Challenge [10, 28] were known to exist, then the impact on information security could hardly be measured.

For an example of a non-cryptographic computation with similar properties, a method of cheating for computing the

permanent of a matrix would imply a complexity-theoretic collapse of classes—specifically that the class BPP equals the class P# [6]. This would be a breakthrough result in the field of complexity theory.

When there is no realistic chance of cheating, the probability distribution can be represented by a constant zero.

Implementing redundancy. The server S allocates a task redundantly as follows. S selects an integer $M \leq N$, the poll size, and a number $\epsilon > 0$, the poll *tolerance*. S then chooses, uniformly at random, a set of M clients from the population and sends them the task. Upon receiving results from the M clients, S puts the output of the computation to a *vote*.

If S chooses to audit, then it submits the task to a trusted client (which may be S itself). Now knowing the correct result, S *penalizes* all clients whose results are incorrect.

If S chooses not to audit, then it looks for a *consensus* in the vote, i.e., a set of $> \frac{1}{2}M + \epsilon$ clients who produce identical results. If there is a consensus, then S accepts the winning result as correct. It then “rewards” the polled clients in the consensus and “penalizes” the others. (Details of “rewards” and “penalties” are spelled out subsequently.) If there is no consensus, then S reassigns the task to a new set of M clients and seeks a new consensus. In common with Golle and Stubblebine [17], we assume that a consensus will eventually be reached. Of course, S stores all client responses, so that when a consensus is finally reached, S knows which clients answered correctly and which did not.

The payment scheme. Let C be the maximum cost incurred by any client when computing any task.² Positing a single constant C does not in any way restrict the possible heterogeneity of either the clients or the tasks in the workload.

We denote by R the *per-task reward* to each client whose result is accepted and by P the *per-task penalty* for each client whose result is not accepted, either because S determines its incorrectness (via an audit) or because the result differs from that of a consensus. To simplify computations, and to ensure that no client loses by participating in the project—and that “good” clients actually profit while “bad” ones do not, we posit certain relationships between C and both R and P . Specifically, we posit the existence of $\rho > 1$ such that $R = \rho C$, and of $\pi > 0$ such that $P = \pi C$. Pragmatism demands certain tighter bounds on ρ and π . Thus, to attract desirable clients, ρ must be large enough a truth-teller can expect a profit. Symmetrically, to deter undesirable clients, π must be large enough a cheater can expect not to receive any profit. Our analyses reflect these desiderata.

3.3 The IAG Protocol

Summing up the development of the IAG, we posit that the server execute the following protocol when allocating a task.

1. Choose a set of M clients from the population uniformly at random and without replacement.
2. Announce α , the probability that it will audit the task.
3. Issue the task to all M clients, collect the results, and determine whether or not to audit.

²Most P2PC computations specify minimum system requirements for each client. We assume that C can be estimated from these requirements.

- (a) If S audits, then it immediately penalizes cheaters and pays truth-tellers.
- (b) If S does not audit, then:
 - i. if there is a consensus, then S pays all members of the consensus and penalizes all non-members;
 - ii. if there is no consensus, then S replays the protocol with this task.

4. TOWARD OPTIMAL STRATEGIES

In this section, we analyze the Internet Auditing Game under three separate scenarios: redundant task allocation with unrestricted collusion among clients, redundant allocation with no collusion among clients, and non-redundant allocation. For each, we determine the minimum frequency of auditing, α , that guarantees correctness, under the assumption of rational clients. We then detail the minimum cost implementation of the optimal strategy under each scenario. An overview of our results is offered in Table 1. Table 2 lists the variables we use throughout this section.

4.1 Redundancy With Unrestricted Collusion

4.1.1 Relating all probabilities

Let T_j be an arbitrary team in the population and let m_j be the number of clients from T_j chosen by S to be in a poll, where $1 \leq m_j \leq M$. Our analyses show that, as the probability of fooling the auditor decreases, the auditing frequency necessary to ensure honesty by clients decreases with it.

LEMMA 4.1. *Letting α_j be the auditing probability chosen by the server, the expected profit of the truth strategy for team T_j , denoted $E_j[\text{truth}]$, satisfies:*

$$E_j[\text{truth}] \geq \alpha_j(m_j R - C) + (1 - \alpha_j)(p_s(m_j R - C) + p_f(-m_j P - C))$$

PROOF. This bound reflects the worst-case scenario for T_j under the truth-telling strategy; T_j tells the truth and is rewarded when S audits, but its clients might be penalized when S does not audit. Since T_j is guaranteed to be in the consensus when S audits, it expects a reward of $m_j R - C$ with probability α_j . In other words, T_j will have one member compute the correct answer at a cost C , and all m_j members will use that answer to get a reward of R . However, with probability $(1 - \alpha_j)$ S does not audit. Since a consensus will eventually be reached, the only event that can result in penalization is a consensus reached by incorrect cheater(s). The probability of this event occurring is at most p_f . In this case, T_j will pay the cost of computing the task C along with being assessed the penalty P . \square

LEMMA 4.2. *The expected profit of the cheat strategy, denoted $E_j[\text{cheat}]$, satisfies:*

$$E_j[\text{cheat}] \leq \alpha_j(p_s(m_j R) - p_f(m_j P)) + (1 - \alpha_j)(m_j R).$$

PROOF. This bound reflects the best case scenario for T_j under the cheating strategy. When S does not audit, T_j gets the reward $m_j R$ without incurring the cost of computing the correct result. Additionally, with probability p_s , T_j gets the reward if S does audit. If S does audit and T_j cheats unsuccessfully, then T_j is penalized. \square

Scenario	Reward	Penalty	Freq. of Audit
Redundancy- Unrestricted Collusion	$\rho > 2$	$\pi > 6$	$\alpha > 5/8$
Redundancy- No Collusion	$\rho > 2$	$\pi > 3/2$	0
No Redundancy	$\rho > 2$	arb. high	arb. low

Table 1: An overview of the conditions for the server’s minimum cost implementation under each scenario.

N	the size of the client population
T_j	the j th colluding team
t_j	the number of clients in team T_j
M	the number of clients in a poll
m_j	the number of members of T_j chosen in a poll
C	the clients’ (common) cost of computing the correct result
$\rho > 1$	the reward constant
R	the per-client, per-task reward: $R = \rho C$
$\pi > 0$	the penalty constant
P	the per-client, per-task penalty: $P = \pi C$
α	S ’s probability of auditing
p_s	the probability of passing an audit while cheating
p_f	the probability of failing an audit while cheating: $p_f = 1 - p_s$

Table 2: The variables used in our analyses.

THEOREM 4.1. *The minimum α_j such that T_j is guaranteed to prefer the truth strategy over the cheating strategy is:*

$$\alpha_j = \frac{1}{2} \cdot \left(1 + \frac{1}{p_f \cdot m_j (\rho + \pi)} \right)$$

PROOF. We compare the expected profit of the best-case cheat action to the expected profit of the worst-case truth action. Using the “profit-equating” method [27], we get:

$$\begin{aligned} & \alpha_j (m_j R - C) \\ & + (1 - \alpha_j) (p_s (m_j R - C) + p_f (-m_j P - C)) \\ & > \alpha_j (p_s (m_j R) - p_f (m_j P)) + (1 - \alpha_j) (m_j R) \end{aligned}$$

Substituting $R = \rho C$ and $P = \pi C$, solving for α_j , and noting that $p_s + p_f = 1$ gives the result. \square

COROLLARY 4.1. *As the probability of cheating p_s decreases, the minimum α_j decreases.*

We henceforth restrict attention to the case where $p_s = p_f = 1/2$, i.e., the probability of cheating successfully is equal to the probability of cheating unsuccessfully for a given task. We consider this simplification as the worst-case scenario that the server would be willing to accept before ceasing to offer client reimbursement. We note that if $p_s > 1/2$, then it would likely be more cost-effective for the server to use an iterated Monte-Carlo algorithm based on cheating (and thereby “cut out the middle-man”). This would be particularly true if the server would be satisfied with a high probability of obtaining the correct answer, rather than certainty. While this simplification represents a very “cheatable” computation, subsequent analysis also measures the effect of a decreased p_s . In particular, we continue to show how the auditing probability decreases when p_s decreases.

In the next Subsection, we place bounds on ρ and π that enable certain conditions on the expected profit of each client strategy.

4.1.2 Implications for budgeting

In this Subsection, we find the minimum reward and penalty levels necessary for the server to ensure a profit for the truth-teller and prevent profit for the cheater. Having the required bounds on the reward constant ρ and the penalty constant π , the server can then determine its minimum-cost implementation of our protocol.

We now determine how big ρ must be to ensure that a truth-teller can expect to get a profit and how big π must be to ensure that a cheater cannot expect to get a profit.

THEOREM 4.2. *If $E_j[\text{truth}] > C$ and $E_j[\text{cheat}] \leq 0$, then $\rho > 2$ and $\pi > 6$.*

PROOF. To obtain the bound on ρ , we set the lower bound of the truth-telling strategy from Lemma 4.1 to be $> C$ and solve for ρ . To obtain the bound on π , given the bound on ρ , we set the upper bound of the cheating strategy from Lemma 4.2 to be ≤ 0 and solve for π . The results directly follow. \square

Theorems 4.1 and 4.2 combine to tell us that the minimum auditing probability is $5/8$. Theorem 4.2 also gives us a minimum cost for untrusted resources (client reimbursement). In order to guarantee that the truth tellers make a profit, the server must offer a per-client, per-task reward of more than $2C$, regardless of the poll size. Therefore, for a minimum-cost implementation, the server will keep poll sizes as small as possible, pay the clients at least $2C$ each, and audit with probability $> 5/8$. A natural question is whether this auditing frequency can be decreased at all so that the cost of trusted resources might decrease. We address this question in the next subsection.

4.1.3 Dominating collusions

In this section, we show that if certain information were known to both server and clients, then the auditing frequency α can be significantly reduced. Toward this end,

suppose the server knows the team sizes, and the teams know M , the poll size, N , the population size, and ϵ , the consensus threshold. Although this information may be unavailable or kept private, our analysis reveals the interesting fact that, as the probability that a single team satisfies a consensus by itself increases, α can be decreased. Of course, we strive to quantify this decrease as well as to expose it. To this end, we introduce the notion of domination. We say that the team T_j *dominates* a poll if $\frac{1}{2}M + \epsilon \leq m_j \leq M$.

LEMMA 4.3. *Given knowledge of M , N , and ϵ , a team T_j can determine the probability θ_j that it will dominate a poll.*

PROOF. Let T_j be an arbitrary team, and let X_j be the random variable representing the team size m_j . Since S selects without replacement, and all $\binom{N}{M}$ ways of selecting the clients in the poll have the same probability, the probability of getting exactly x clients from T_j is:

$$\Pr[X_j = x] = \binom{t_j}{x} \binom{N - t_j}{M - x} \div \binom{N}{M}.$$

Thus, X_j has a hypergeometric distribution. θ_j is determined by directly calculating $\Pr[X_j \geq \frac{1}{2}M + \epsilon]$. \square

THEOREM 4.3. *If team T_j dominates the poll with probability θ_j , then the smallest audit frequency α_j under which T_j is guaranteed to prefer the truth action over the cheat action is:*

$$\alpha_j = \frac{1}{1 + p_f - \theta_j} \cdot \left((1 - \theta_j) + \frac{1}{m_j(\rho + \pi)} \right)$$

PROOF. Lemma 4.3 allows us to rewrite Lemma 4.1 with θ_j , the probability of domination, as follows:

$$\begin{aligned} E_j[\text{truth}] &\geq \\ &\alpha_j(m_j R - C) + (1 - \alpha_j) \cdot \\ &(\theta_j(m_j R - C) + (1 - \theta_j)(-m_j P - C)) \end{aligned}$$

Reconstructing the proof of Theorem 4.1 with this new bound yields the result. \square

We see that T_j 's expected profit for truth-telling increases with θ_j . The intuition behind this is that the average cost of computation per team member is C/m_j , which decreases as m_j increases. Therefore, larger teams actually profit more under the truth-telling strategy than smaller ones, but their upper bound on expected profit from cheating remains the same.

COROLLARY 4.2. *If some T_j dominates the poll with very high probability, then the minimum value of α_j approaches $2/(M \cdot (\rho + \pi))$ (from above).*

PROOF. As θ_j approaches 1, m_j approaches M . The rest follows from our simplification that $p_f = 1/2$. \square

We stress that this bound cannot be realized unless the team sizes were known by the server and the poll size, population size, and consensus threshold were known by the teams.

4.2 Redundancy With No Collusion

We now consider what happens when the server can prevent collusion among clients. We assume that the approximate probability of a cheater returning a result different from the poll consensus is independent and bounded below by $(1 - p_s)$. It becomes clear by the end of this section that even a small decrease in p_s will have a substantial effect on reducing the frequency of auditing.

Say that $t_j = 1$ for all T_j , i.e., that there is no collusion. Recalling that the consensus threshold ϵ is chosen by the server, we consider the following result.

THEOREM 4.4. *Say that $p_s = 1/2$. If ϵ is polynomial in M , then the probability that the consensus has the incorrect answer is $\leq e^{-\epsilon^2/M}$.*

PROOF. Since cheating and truth-telling are the only two strategies under consideration, an incorrect outcome can occur only when a consensus is comprised of (non-colluding) cheaters. So suppose that all M clients are cheaters, which represents the event in which such a consensus is most likely. Let X_j be a random variable indicating that the j th client cheats with a unique result, and let $X = \sum_{j=1}^M X_j$. Since this event occurs independently with probability at least $(1 - p_s)$, we use a Chernoff bound to bound the probability that an incorrect consensus occurs. We find that

$$\Pr[X \leq \frac{1}{2}M - \epsilon] \leq e^{-(2\epsilon)^2(1-p_s)/2M} = e^{-\epsilon^2/M}.$$

\square

In the light of Theorem 4.4, we reexamine the bounds on α by reexamining the worst-case reward for the truth-teller and the best-case reward for the cheater. For the duration of this subsection, let $\epsilon = \sqrt{kM}$ for some $k > 0$.

LEMMA 4.4. *Say that the population of clients is collusion-free. The expected profit of the truth-telling strategy satisfies:*

$$\begin{aligned} E[\text{truth}] &\geq \\ &\alpha(R - C) + (1 - \alpha) \cdot \\ &\left((1 - e^{-k})(R - C) + e^{-k}(-P - C) \right) \end{aligned}$$

The expected profit of the cheating strategy satisfies:

$$\begin{aligned} 2 \cdot E[\text{cheat}] &\leq \\ &\alpha(R - P) + (1 - \alpha) \cdot \\ &\left(R + e^{-k} \cdot R - (1 - e^{-k}) \cdot P \right) \end{aligned}$$

PROOF. For the bound on $E[\text{truth}]$, we proceed as in Lemma 4.1. Observe that a truth-teller risks incurring a penalty only when the consensus is incorrect, which occurs with probability $\leq e^{-k}$ (where $k = \epsilon^2/M > 0$) by Theorem 4.4. For the bound on $E[\text{cheat}]$, observe that a cheater has probability of at most $1/2$ of passing an audit. If the server does not audit, there are two cases: a correct consensus, and an incorrect consensus arrived at by other cheaters. If the consensus is correct, then the cheater has a reward probability of $p_s = 1/2$. If the consensus is incorrect, then we incorporate the bound from Theorem 4.4 for the expected reward and penalty amounts. \square

THEOREM 4.5. *The smallest α under which T_j is guaranteed to prefer the truth-telling strategy over the cheating strategy is*

$$\alpha = \max \left\{ 0, 1 - e^k \cdot \left(\frac{1}{3} - \frac{2}{3(\rho + \pi)} \right) \right\},$$

where $k = \epsilon^2/M > 0$.

PROOF. By the profit-equating method, we set the lower bound for the truth-telling strategy from Lemma 4.4 to be strictly greater than the upper bound for the cheating strategy from Lemma 4.4 and then solve for α . Assuming $(\rho + \pi) > 7/2$ (see Section 4.2.1 for the proof), this process yields an expression that becomes negative for sufficiently large M . Since probabilities must be nonnegative, this switch in sign occurs precisely when the lower bound on the expected profit of the truth-telling strategy has surpassed the upper bound on the cheating strategy. We interpret this as mandating that the auditing probability becomes zero. \square

Since $k = \epsilon^2/M$, and ϵ is polynomial in M , Theorem 4.5 shows that the decrease in the auditing probability is exponential in M . Further, as the probability p_s decreases, so also does the Chernoff bound in the proof of Theorem 4.4. Consequently, Theorem 4.5 shows that α decreases in this case also.

4.2.1 Implications for budgeting

We now show that, while the reward necessary to ensure that every truth-teller profits does not change from the case of unrestricted collusion, the minimum penalty required to deprive the cheater of income decreases to $3/2$. We also show how α vanishes for fairly low values of M .

We proceed as in Section 4.1.2.

THEOREM 4.6. *If $E_j[\text{truth}] > C$, then $\rho > 2$. If, in addition, $E_j[\text{cheat}] \leq 0$, then $\pi > 3/2$ for all sufficiently high poll sizes.*

PROOF. Using the bound on $E_j[\text{truth}]$ of Lemma 4.4, and noting that $\alpha_j \leq 1$, it follows that $\rho > 2$. Further, using the bound on $E_j[\text{cheat}]$ of Lemma 4.4 and noting that $\rho > 2$ and e^{-k} tends to zero, it follows that $\pi > 3/2$. \square

We have done simulations to find how large a poll size has to be before auditing is no longer required—because α has vanished. We consider two values for ϵ : $\frac{1}{2}M$ (unanimous consensus) and $\frac{1}{4}M$ (where ϵ is middle-valued). Table 3 illustrates this minimum poll size, call it M_{min} , for both values of ϵ .

Note that even an order of magnitude increase in π does not necessarily have a proportionate effect on M_{min} . Thus, the server does not have to impose harsh penalties in order to prevent cheating.

Finally, we find that the minimum cost implementation depends mostly on the choice of the consensus threshold ϵ . Here, the cost of trusted resources vanishes for fairly small poll sizes with a unanimous consensus, but the poll size increases as the consensus size decreases. As examples, the minimum total cost for unanimous consensus where the penalty is equal to the reward (i.e., $\pi = 2$) is more than $14C$, while the minimum cost implementation for a $3/4$ consensus with the same penalty level is greater than $58C$.

4.3 Non-redundant Allocation

We now consider non-redundant allocation and determine the minimum auditing frequency.

4.3.1 Relating all probabilities

Now we consider the protocol when tasks are assigned to single clients. The notion of consensus becomes irrelevant in this case: if the server does not audit, then the client still gets the reward. However, the truth-telling and cheating strategies are still relevant. We can now calculate the exact expected profits for these strategies, to get a precise bound on α . The following results yield to our earlier techniques, so are left to the reader.

LEMMA 4.5. *When tasks are allocated non-redundantly, then*

$$E[\text{truth}] = R - C,$$

and

$$E[\text{cheat}] = \alpha(p_s R - p_f P) + (1 - \alpha)R.$$

THEOREM 4.7. *Suppose $\rho > 2$. The minimum α such that a client is guaranteed to prefer truth-telling over cheating is*

$$\alpha = \frac{1}{p_f(\rho + \pi)}$$

We see that as the probability of cheating decreases, so too does the auditing probability α . If the probability of cheating is fixed, then α depends solely on ρ and π . In the next section, we determine the reward and penalty levels necessary to ensure that only truth-tellers achieve a profit.

4.3.2 Implications for budgeting

We proceed as in Section 4.1.2.

THEOREM 4.8. *If $E_j[\text{truth}] > C$, then $\rho > 2$. If, in addition, $E_j[\text{cheat}] \leq 0$, then $\pi > 0$.*

PROOF. We invoke Lemma 4.5 for the bounds on ρ and π . \square

For any implementation, the server must pay more than $2C$ to each client in order to enable the truth-teller to make a profit. The auditing probability α will be less than 1 in such a case, but can be made arbitrarily low with an arbitrarily high π . The server does not have to be concerned with an unfair penalty, because Lemma 4.5 shows that the truth-teller is never penalized. On the other hand, π must be nonzero. Otherwise, a rational client will never tell the truth. Therefore, the minimum cost implementation has a low (albeit positive) cost of trusted resources and a cost of untrusted resources above $2C$.

5. APPLYING THE IAG

In this section, we describe an application of the IAG in order to demonstrate its practical significance. Toward this end, we use parameters based on Gray’s study of P2PC costs [19]. Consider a fictitious entity “X” that is launching a SETI-like P2PC project, where each task requires approximately 8 CPU hours on a 3GHz machine to complete. In addition, the probability of cheating is negligible, the poll consensus is unanimous, and auditing is computationally intensive. We suppose that a client base collectively agrees

$$\epsilon = \frac{1}{2}M:$$

π	1.5	3	5	10	100	1000+
M_{min}	8	7	6	6	5	5

$$\epsilon = \frac{1}{4}M:$$

π	1.5	3	5	10	100	1000+
M_{min}	32	26	23	21	18	18

Table 3: Illustrating M_{min} , the minimum poll size where auditing is no longer required. In each table, M_{min} corresponds to the values of the penalty constant π under the given consensus threshold ϵ . The reward constant ρ was fixed above 2.

Parameter	Non-Redundant	Redundant with Collusion	Redundant w/o Collusion
Minimum Poll Size	1	3	2
Penalty	\$49.50	\$1.50	\$0.50
Expected Auditing Frequency	0.5%	62.5%	0%
$Cost_{untrusted}$	\$0.50	\$1.50	\$1.00

Table 4: A sample of the per-task parameters obtained by applying the IAG to a SETI-like P2PC.

to participate (rationally) at a reward of \$0.50 per correct task, which is a substantial discount from the per-task cost of new hardware. Luckily, X already has a server system at its disposal, but that system cannot audit more than 1% of the tasks it allocates. Company “Y”, upon learning X’s situation, offers a centralized identification mechanism that prevents collusion among clients. The question becomes: should X pay for this mechanism? If so, how much?

Our model answers these questions precisely. Based on results from the previous section, Table 4 gives parameters for the three task allocation scenarios, as well as the resulting auditing frequency and the per-task cost of untrusted resources. Clearly, non-redundant allocation is less expensive than the two alternatives, regardless of the number of tasks, since the existing hardware is capable of performing the auditing under this scenario. Thus, X should reject Y’s offer.

Next we consider the same P2PC, but increase the cheating probability to 1/2. In the case of redundancy with no collusion, results from Section 4.2 show that the poll size, penalty, and $Cost_{trusted}$ must be increased to 6, \$1.25, and \$3.00 respectively in order for the auditing frequency to remain zero. However, the auditing frequency for non-redundant allocation increases only to 1%. Since the existing hardware is still capable of all auditing, X should again reject Y’s offer and proceed with non-redundant task allocation.

These simple examples illustrate how our model can be applied as a heuristic for intelligent budgeting and design choices. Our model can also be used to estimate how much work can be done given a fixed budget, as well as calculating the expected effects of various penalty and reward parameters for an initial launch of a project.

6. CONCLUSIONS

Our goal was to estimate the cost of redundancy in commercial P2PC using the *Internet Auditing Game* defined in Section 3. Toward this end, we modeled three distinct task allocation scenarios that a server might face when organizing a real P2PC commercial project. We first examined the cost of implementing redundancy when collusion among clients is unrestricted- which is a possible scenario when the clients are drawn from the Internet. In our idealized model, we proved that the server must allocate a substantial amount

to trusted resources for auditing purposes. An interesting consequence of our analysis occurs when a single colluding group is known to dominate a poll. In this case, the amount of auditing required to guarantee correctness (and consequently, the cost of trusted resources) drops significantly. This suggests that the server is no worse off by fostering collusion if it already exists in the population.

We also found that redundancy can eliminate the need for auditing (and thus the cost of trusted resources) when collusion is prevented. Further, we found that poll sizes remain small in this case for a unanimous consensus, but increase for lower consensus sizes. However, without a costly centralized authority to certify identities, it may be practically impossible to eliminate collusion due to the lingering potential for a Sybil attack [9]. We conclude that further investigation into resisting the effects of collusion, especially simple low-cost measures suitable for P2PC projects, is clearly worthwhile.

Finally, we found that non-redundant task allocation offers a cost-effective alternative to redundancy, especially when collusion is possible. Here, the cost of trusted resources becomes arbitrarily low based on a penalty that cannot be levied against honest clients. Also, the cost of untrusted resources is lower than those of any scenario with redundancy, since only one client is employed per task.

In order to demonstrate the practical significance of our model, we offered an application to a P2PC project that closely resembles SETI@home. We believe that additional applications of the model, as well as extensions, will come with further research.

6.0.2.1 Acknowledgment.

The research of A. Rosenberg and M. Yurkewych was supported in part by NSF Grant CCF-0342417. The research of B.N. Levine was supported in part by NSF Grant ANI-0133055. The authors wish to thank George Bissias, Louis Theran, and an anonymous CCS reviewer for helpful comments.

7. REFERENCES

- [1] D. Anderson. BOINC: A System for Public-Resource Computing and Storage. *5th IEEE/ACM Intl. Wkshp. on Grid Computing*, 2004.
- [2] D. Anderson, J. Cobb, E. Korpela, Matt Lebofsky, D. Werthimer. SETI@home: An Experiment in

- Public-Resource Computing. *Comm. ACM* 45, 2002.
- [3] Berkeley Open Infrastructure for Network Computing (BOINC). <http://boinc.berkeley.edu>
- [4] D. Bernstein. Protecting Communications Against Forgery. *Algorithmic Number Theory*, J. Buhler and P. Stevenhagan (Ed.), to appear 2005.
- [5] S. Buchegger, J. Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes - Fairness in Distributed Ad-hoc Networks. *IEEE/ACM Wkshp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2002.
- [6] J-Y. Cai, A. Nerukar, D. Sivakumar. On the Hardness of Permanent. *Proc. 16th Ann. Symp. on Theoretical Aspects of Computer Science*, 1999.
- [7] L. P. Cox, C. D. Murray, B. D. Noble. Pastiche: Making Backup Cheap and Easy. *5th ACM/USENIX Symp. on Operating Systems Design and Implementation*, 2002.
- [8] R. Dingleline, N. Mathewson, P. Syverson. Tor: The Second-Generation Onion Router. *13th USENIX Security Symp.*, 2004.
- [9] J. Douceur. The Sybil Attack. *1st Intl. Wkshp. on Peer-to-Peer Systems*, 2002.
- [10] Distributed.net. <http://www.distributed.net>
- [11] EBay. <http://www.ebay.com>
- [12] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. *6th Intl. Wkshp. on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2002.
- [13] FightAIDS@home. <http://www.fightaidsathome.com>
- [14] Folding@home. <http://folding.stanford.edu>
- [15] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. *2nd Intl. Wkshp. on Peer-to-Peer Systems*, 2003.
- [16] P. Golle and I. Mironov. Uncheatable Distributed Computations. *Topics in Cryptology, CT-RSA*, 2001.
- [17] P. Golle and S. Stubblebine. Secure Distributed Computing in a Commercial Environment. *Proceedings of Financial Cryptography*, 2001.
- [18] The Google Compute Project. <http://www.toolbar.google.com/dc/offerdc.html>
- [19] J. Gray. Distributed Computing Economics. Microsoft Research TR-2003-24, March 2003.
- [20] R. Hogg and E. Tanis. *Probability and Statistical Inference*, 4th Ed. Macmillan Publishing, 1993.
- [21] U. Maurer. Information-Theoretic Cryptography. *Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science*, Springer-Verlag, vol. 1666, 47-64, 1999.
- [22] J. Ledlie, J. Shneidman, M. Seltzer, J. Huth. Scooped, Again. *2nd Intl. Wkshp. on Peer-to-Peer Systems*, 2003.
- [23] D. Mookherjee, I. Png. Optimal Auditing, Insurance, and Redistribution. *Quarterly J. Economics* 104, 399-415, 1989.
- [24] J.F. Nash. Equilibrium Points in N-Person Games. *Proc. National Academy of Sciences of the United States of America* (36), 1950.
- [25] J.F. Nash. Non-Cooperative Games. *Annals of Mathematics* (54), 1951.
- [26] M. Osborne, A. Rubenstein. *A Course in Game Theory*. MIT Press, 1994.
- [27] E. Rasmusen. *An Introduction to Game Theory*, 3rd Ed. Blackwell Publishing, 2003.
- [28] RSA Secret Key Challenge. <http://www.rsasecurity.com/rsalabs/node.asp?id=2100>
- [29] K. Sanzgiri, B. Dahill, D. LaFlamme, B. N. Levine, C. Shields, E. Belding-Royer. A Secure Routing Protocol for Ad hoc Networks. *IEEE/ACM Journal of Selected Areas in Communications: Special issue on Wireless Ad hoc Networks (JSAC)*, 2005.
- [30] J. Shneidman and D. Parkes. Rationality and Self-Interest in Peer to Peer Networks. *2nd Intl. Wkshp. on Peer-to-Peer Systems*, 2003.
- [31] J. Shneidman and D. Parkes. Using Redundancy to Improve Robustness of Distributed Mechanism Implementations. *4th ACM Conf. on Electronic Commerce*, 2003.