

Accounting Mechanisms for Distributed Work Systems

Sven Seuken
School of Engineering
& Applied Sciences
Harvard University
Cambridge, MA

seuken@eecs.harvard.edu

Jie Tang
Computer Science Division
University of California at
Berkeley
Berkeley, CA

jietang@eecs.berkeley.edu

David C. Parkes
School of Engineering
& Applied Sciences
Harvard University
Cambridge, MA

parkes@eecs.harvard.edu

ABSTRACT

A significant challenge in distributed work systems is to address the moral hazard problem wherein users may seek to free-ride on the work contributed by others. We formalize the problem of designing *incentive-compatible accounting mechanisms*, that measure the net contributions of users despite relying on voluntary reports of work performed and work received. A new mechanism is introduced that removes any incentive for a user to manipulate via misreports, about either work contributed or work consumed. The mechanism is demonstrated in simulation to provide good system efficiency compared to an existing, manipulable mechanism. In closing, we illustrate that sybil attacks are, however, powerful in accounting mechanisms which leaves an open research challenge.

1. INTRODUCTION

The Internet connects millions of people and leads to novel ways of interaction and collaboration. Distributed work systems arise in many places where individual users perform work for other users, often called peer production [3]. For example, users of a peer-to-peer (P2P) file-sharing network share videos, music or software with each other. The users of a P2P backup system provide their hardware space to each other to backup each other's data. Amazon Mechanical Turk and other so-called "crowd sourcing" applications suggest an explosion of interest in new paradigms for economic production.

Of course, the total work performed by a user population must equal the total work consumed. Moreover, while some degree of free-riding may be acceptable (e.g., if some users are altruistic while work is extremely costly for others), it is generally accepted that the long-term viability of work systems that operate without the use of monetary transfer must rely on roughly balanced work contributions by the great majority of users. Often time this is achieved by seeking to enforce temporally-local balance, such as via a fair exchange protocol such as BitTorrent [6]. Yet, this "local balance" clearly introduces a large inefficiency—users are limited to consuming work at a rate at which they can themselves produce work, must be able to simultaneously consume and produce work, and cannot perform work and store credits for future work consumption [11].¹

¹Pouwelse et al. [12] have found that more than 80% of BitTorrent

The problem of accounting mechanisms is motivated by this established need to tally work contributed and work consumed, so as to be able to improve system efficiency by temporally decoupling these two activities. But the challenge in designing a useful accounting mechanism is that it must work despite relying on voluntary reports by the same users that may seek to free-ride. For this reason, it needs to be robust to manipulations; e.g., to improve my standing in the system I might seek to overstate my amount of work contributed, or understate my amount of work consumed.

1.1 Reputation vs. Accounting Mechanisms

The problem of designing an incentive-compatible accounting mechanism shares some features with work on transitive reputation mechanisms [8, 1, 4, 5, 14, 2], in that if A trusts B enough to perform some work for B and B trusts C enough to perform some work for C then A should also have some level of trust in C . Yet, there are some significant differences. First, and somewhat informally, the essence of accurate accounting is the operation of *addition* whereas the essence of accurate reputation aggregation is the operation of *averaging*. Second, in distributed work systems, every positive report by A about his interaction with B , i.e., B performed work for A , is simultaneously a negative report about A , i.e., A received work from B . This fundamental tension is not present in reputation mechanisms where a good report by A about B does not reflect badly on A .

Third, sybil attacks in which a user creates multiple fake identities are more powerful in distributed work systems. In PageRank for example, the only concern about a sybil attack is that a user can increase the reputation (i.e. rank) of his website by creating a set of sybils that are linking to his original website. A user does not directly benefit from a sybil website with a high reputation. Various reputation mechanisms (e.g., maxflow, hitting-time, shortest-path [4, 14, 1]) are sybil-proof in this sense. The situation is drastically different in distributed work systems. For example, in a P2P file-sharing network, if I can create sybils with some positive reputation, then I can use these sybils to receive work from other users without having the negative effect of having that "received work" be associated with my real account. Standard sybil-proof mechanisms do not protect against these kinds of sybil attacks and thus a new approach is required.

We are interested also in *decentralized* accounting mechanisms in which there is no central infrastructure to collect reports about work performed and consumed and run information-aggregation algorithms. Rather, in systems such as P2P file sharing-networks it is interesting to strive for fully distributed architectures.

users go offline immediately once they have finished downloading. Long-term accounting mechanisms would solve this problem by giving users an incentive to share even after they have finished downloading.

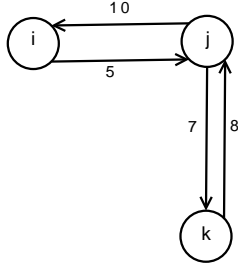


Figure 1: A sample work graph. All edges have one weight.

1.2 Overview of Results

In this paper, we present the first formal model of the problem of incentive-compatible accounting mechanisms and introduce the *Drop-Edge Mechanism*, a new mechanism that is robust to mis-reports. Along the way, we suggest a simple attack of *BarterCast* [10], an existing accounting mechanism. The simulation results suggest that the effect of misreports in BarterCast can be significant, allowing users to successfully free-ride and reduce the efficiency of the system for cooperating users. Our new mechanism, in comparison achieves much better system efficiency. Finally, we consider the question of sybil attacks and show multiple ways in which BarterCast as well as our mechanism can be manipulated via sybils. This leaves to an interesting open research challenge.

1.3 Related Work

The mechanism that we present is motivated by BarterCast [10], a max-flow based decentralized accounting mechanism for P2P file-sharing networks. BarterCast was introduced by Meulpolder et al. to successfully distinguish between cooperative peers and free-riders, and to improve the efficiency for cooperative peers over time. However they do not analyze the attacks presented here in detail. Piatek et al. [11] also study the problem of decentralized accounting in P2P file-sharing, and find empirically that most users of a P2P network are connected via a one hop link in the connection graph. Based on this, they propose to use well-connected intermediaries to broker information on contributions by users. The main drawback of this approach is the special role that the intermediaries play because they become bottlenecks for the system and furthermore there is no incentive for them to behave truthfully in serving as intermediaries. A different approach to accounting in P2P system is taken by the literature on virtual currency (scrip) systems [7]. Our work is very different from this because we do not require a trusted currency, and work is not intermediated by a trusted center.

2. FORMAL MODEL

2.1 Distributed Work Systems

Consider a distributed work system of n agents each capable of doing work for each other. For example, this could be P2P file sharing in which work is done by uploading a file from one peer to another. All work is assumed to be quantifiable in the same units. The work performed by all agents in the system is captured by a work graph (see Figure 1):

Definition 1. (Work Graph) A work graph $G = (V, E, w)$ has vertices $V = \{1, \dots, n\}$, one for each agent, and directed edges $(i, j) \in E, i, j \in V$ corresponding to work performed by i for j , with weight $w(i, j) \in \mathbb{R}_{\geq 0}$ denoting the number of units of work.

In general, the work graph is unknown to individual agents because it represents the true work performed by all agents. Instead, every agent only has direct information about its own participation:

Definition 2. (Agent Information) Each agent $i \in V$ keeps a private history $(w_i(i, j), w_i(j, i))$ of its direct interactions with

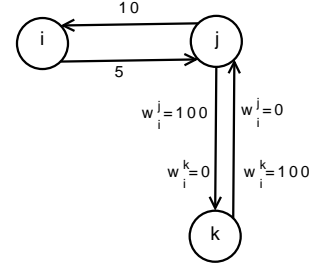


Figure 2: A subjective work graph from agent i 's perspective. Edges where i has direct information have only one weight. Other edges can have two weights, corresponding to the reports of the two agents involved.

other agents $j \in V$, where $w_i(i, j)$ and $w_i(j, i)$ are the work performed for j and received from j respectively.

Whereas the work system itself is always distributed and is not intermediated, we consider both centralized and decentralized accounting mechanisms. In both cases, agents are able to make voluntary (and possible untruthful) reports about their local information. Based on its own experiences and known reports from other agents, agent i can construct a subjective work graph (see Figure 2):

Definition 3. (Subjective Work Graph) A subjective work graph (from agent i 's perspective), $G_i = (V_i, E_i, w_i)$, is a set of vertices $V_i \subseteq V$ and directed edges E_i . Each edge $(j, k) \in E_i$ for which $i \notin \{j, k\}$, is labeled with one, or both, of weights $w_i^j(j, k), w_i^k(j, k) \in \mathbb{R}_{\geq 0}$, denoting the edge weight as reported by agent j and agent k respectively. For edges (i, j) and (j, i) the associated weight is $w_i^i(i, j) = w(i, j)$ and $w_i^i(j, i) = w(j, i)$ respectively.

Note that $w_i^j(j, k)$ and $w_i^k(j, k)$ need not be truthful reports about $w(j, k)$. In a centralized accounting mechanism, agents can make claims about work to the center, which then shares these reports with all other agents. In a decentralized mechanism, these reports are "gossiped" over the system from agent to agent. The suggested exchange protocol that an agent follows, potentially subject to alteration by a manipulating agent, is:

Definition 4. (Information Exchange Protocol) Whenever agent i is contacted by agent j via the gossip protocol, the agents exchange information about work performed and received in the N_r most recent transactions and also about the N_h agents from which the most work has been received.

2.2 Accounting Mechanisms

In distributed work systems, every agent i who is willing to perform work might receive a work request by a set of agents that he has rarely or never interacted with before. This induces a choice set:

Definition 5. (Choice Set) We let $C_i^t \subseteq V \setminus \{i\}$ denote the choice set for agent i at time t , i.e., the set of agents that are interested in receiving some work from i at time t .

We assume that an agent has no *a priori* bias towards assisting one agent over another. Furthermore, it is not useful for an agent to receive work from or perform work for one of its own sybil nodes. The role of an accounting mechanism is to provide an estimate of the net work contributed by each agent $j \in C_i^t$ to the system.

Definition 6. (Accounting Mechanism) An accounting mechanism takes as input a subjective work graph G_i , a choice set C_i^t , and determines the reputation $R_j(G_i, C_i^t)$, for any agent $j \in C_i^t$, as viewed by agent i .

Based on the results of the accounting mechanism, the agent then decides which actions to take, i.e., how much work to perform for which agent.² We will use the following allocation method in this paper:

Definition 7. (Winner-takes-all Allocation) Given subjective work graph G_i and choice set C_i^t , agent i performs one unit of work for agent $j \in \arg \max_{k \in C_i^t} R_k(G_i, C_i^t)$, breaking ties at random.

In general, other allocation rules can be used, e.g., proportional allocation, or threshold rules, and we will consider those rules for future research. However, in this paper we focus on the winner-takes-all allocation rule.

2.3 Strategic Manipulations

The basic concern in distributed work is to prevent users from free riding on the contributions of others. For this purpose, we adopt the model of Meulpolder et al. [10] considering a population that consists of a mixture of *cooperative* agents, who always offer one unit of work in every time step, and *malicious* agents (or so-called “lazy free-riders” [10]) who try to intermittently shirk work. The role of an accounting mechanism is to make it difficult to benefit by being a malicious agent. But given the presence of an accounting mechanism, we also model a subset of the malicious agents as *strategic* agents that will try to manipulate the mechanism in various ways while also trying to free ride.³

Manipulations of the accounting mechanism can occur by misreports of work, both performed and consumed, and also by introducing sybil nodes that may perform and consume work while also misreporting information. We suggest the following ranking of possible kinds of manipulations (in order of decreasing importance): (1) misreports to increase your own reputation; (2) misreports to decrease someone else’s reputation; (3) sybil attack to increase your own or your sybil’s reputation; (4) sybil attack to decrease someone else’s reputation.

Misreports are easier to execute than sybil attacks because a user must simply modify the information reporting protocol of a client (to either the center, or via gossiping, depending on whether the mechanism is centralized or decentralized.) On the other hand, successful sybil attacks require adopting multiple IP addresses and therefore using anonymizing networks such as Tor, preventing an additional level of complication. Moreover, if work were performed or received under one of these sybils then this would need to be coordinated with the “main” user application. In addition, attacks that boost yourself are more useful than attacks to hurt another user because the former provides a user with an increased reputation in every interaction, whereas decreasing another user’s reputation only benefits the user in the event that the other user is in the same choice set (which will occur with low probability as the network size increases).⁴ We believe that this prioritization provides useful guidance in developing an accounting mechanism that

²In adopting the term “reputation” for the output of the accounting mechanism we are consistent with prior work [10], but not because we intend to convey any additional similarity between accounting and reputation mechanisms.

³Note that this simple model, in which cooperative agents contribute work every period is designed as a simple test of the effectiveness of an accounting mechanism and is not meant to imply that cooperative behavior requires that an agent contributes in each and every period!

⁴Note that we assume that the way agents make it into one another’s choice set is not subject to manipulation. Thus, sybil attacks where sybils are used to increase the number of interaction partners are not explicitly considered [9]. Furthermore we don’t consider attacks on the amount of information gossiped (via the suggested information exchange protocol) because less information is not useful when attacking, and we assume that other agents will simply ignore additional information. Last, we do not consider manipulations that increase the frequency with which information is spread.

strikes a good balance between informativeness and robustness to manipulation. For the most part, we will focus in what follows on misreport manipulations:

Definition 8. (Misreport Manipulation) A misreport manipulation is one in which agent i reports untruthful information about its work in the system, either to the center in a centralized accounting mechanism or via an information exchange protocol in a decentralized accounting mechanism.

Given this, we define an accounting mechanism that is robust to manipulation as follows:

Definition 9. (Misreportproof) An accounting mechanism is misreportproof if, for any agent $i \in V$, any subjective work graph G_i , any work graph G , and any choice set C_i^t in period t , no agent $j \in C_i^t$ has a misreport manipulation for which:

- $R_j(G'_i, C_i^t) > R_j(G_i, C_i^t)$,
- or $R_k(G'_i, C_i^t) < R_k(G_i, C_i^t)$ for some $k \in C_i^t \setminus \{j\}$,

where G'_i is the subjective work graph of agent i induced by the misreports.

This condition is valid to preclude a useful misreport manipulation by any agent that has non-negative utility for consuming work of others, non-positive utility for performing work, and when coupled with an allocation rule that allocates (weakly-) monotonically less work to agent i as its reputation decreases and the reputation of other agents increases. Given an accounting mechanism that is misreportproof, we assume that users who consider only misreport attacks will then choose to faithfully follow the intended protocol and be truthful (because reporting truthful information cannot be harmful to themselves).

3. THE BARTERCAST MECHANISM

To the best of our knowledge, the BarterCast mechanism [10] was the first decentralized accounting mechanism to use a decentralized message exchange protocol to build up a subjective bandwidth graph. The mechanism is already implemented in the BitTorrent client *Tribler* [13] and can be downloaded for free.

Definition 10. (BarterCast Mechanism) Given subjective work graph G_i and choice set C_i^t , construct the modified graph $G'_i = (V_i, E_i, w_i)$ with weights defined as (where missing reports are set equal to ∞):

$$\forall (j, k) | i \in \{j, k\} : w'_i(j, k) = w_i(j, k) \quad (1)$$

$$\forall (j, k) | i \notin \{j, k\} : w'_i(j, k) = \min\{w_i^j(j, k), w_i^k(j, k)\} \quad (2)$$

Let $MF(i, j)$ denote the maximum flow from i to j in G'_i . Define the reputation of agent j as $R_j(G_i, C_i^t) = MF(j, i) - MF(i, j)$.⁵

In BarterCast, an agent takes its own information over reports from others and given two reports takes the minimum of the two. By taking the minimum, one agent cannot grossly inflate the work that it has performed for another agent when that other agent also submits a report. On the other hand, one agent can grossly reduce the work that it has consumed (or equivalently, that another agent has contributed.) The motivation for running a max-flow algorithm on the resulting subjective work graph comes from the literature on sybil-proof reputation mechanisms (see, e.g., Cheng and Friedman [4]). The max-flow algorithm bounds the influence of any report that agent j can make by the edges between agents i and j .

⁵This specification of BarterCast differs from Meulpolder et al. [10] only in that they take the arctan of the difference between the flows. Because arctan is a monotonic function this does not change the ranking of the agents and thus it doesn’t make a difference for the experiments in this paper.

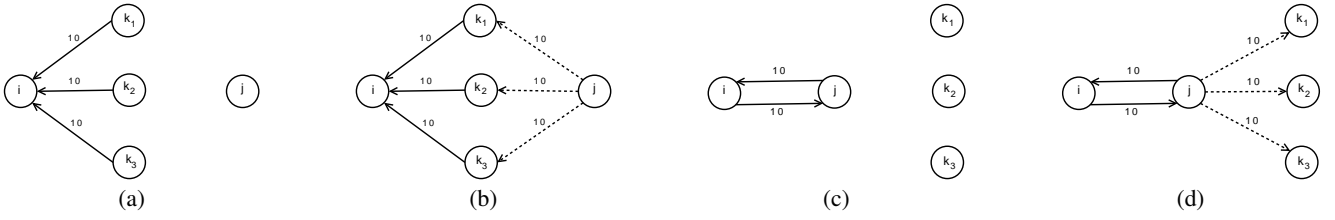


Figure 3: (a) Work graph 1 based on true reports. (b) Subjective work graph as viewed by i , including a misreport attack by j to boost his reputation in BarterCast. (c) Sample work graph 2 based on true reports. (d) Subjective work graph 2 as viewed by i , including a misreport attack by j to decrease agents' k_1, k_2 , and k_3 reputations. Dotted edges indicate misreports.

The BarterCast mechanism can be manipulated in two simple ways via misreports. In fact, *it is a dominant strategy in BarterCast for a strategic agent to always report ∞ work performed for all other agents while having received 0 work*. The informativeness of the mechanism must therefore rely on cooperative, or malicious but non-strategic agents.

See Figure 3. In all cases we are showing the subjective work graph from i 's perspective and the manipulating agent is j . Graph (a) is showing a work graph based on true reports only. Graph (b) is showing the same graph but now including a misreport by agent j to boost its own reputation. Agent j has simply reported that it has done a lot of work for the agents k_1, k_2 , and k_3 . The BarterCast mechanism does not catch this, because there are no reports from these other agents. The max-flow algorithm determines that the maximum flow from j to i is now 30, while the maximum flow from i to j is 0. Thus, using this misreport attack, agent j has increased his reputation from agent i 's perspective from 0 to 30. A second misreport attack is displayed in Figure 3 (c) and (d). Graph (c) shows a new work graph based on true reports only. Graph (d) shows a misreport manipulation by agent j where he simply reported that he did a lot of work for agents k_1, k_2 , and k_3 . This does not affect the reputation of agent j , but it decreases the reputation of agents k_1, k_2 , and k_3 from 0 to -10. Thus, without the manipulation all agents had a reputation of 0 and after the manipulation, agent j has influenced the reputation of the other agents to be negative.

4. THE DROP-EDGE MECHANISM

While the authors of BarterCast [10] argue that only few agents are actually behaving strategically in P2P file-sharing networks, we should expect that this will only be the case if agents can only benefit little from manipulations. Using an accounting mechanism as easily manipulable as BarterCast may open up a large opportunity for successful free-riding (coupled with an ineffectual accounting mechanism.) The experience with *Kazaa Lite* has shown that a majority of users might adopt a hacked client (and thereby become a strategic agent) if the expected benefit from doing so is high.

It turns out that the solution to the misreport manipulations of BarterCast is amazingly simple yet powerful. We use the main paradigm in mechanism design, namely that an agent's own report shall not affect his own "opportunity set", i.e., the set of possible outcomes it can achieve.

Definition 11. (Drop-Edge Mechanism) Given subjective work graph G_i and choice set C_i^t , construct the modified graph $G'_i = (V_i, E'_i, w'_i)$ with E'_i initialized to E_i and the weights defined as (where missing reports equal 0):

$$\forall (j, k) | i \in \{j, k\} : w'_i(j, k) = w_i^t(j, k) \quad (3)$$

$$\forall (j, k) | j, k \in C_i^t : w'_i(j, k) = 0 \quad (4)$$

$$\forall (j, k) | j \in C_i^t, k \notin C_i^t : w'_i(j, k) = w_i^k(j, k) \quad (5)$$

$$\forall (j, k) | k \in C_i^t, j \notin C_i^t : w'_i(j, k) = w_i^j(j, k) \quad (6)$$

$$\forall (j, k) | j, k \notin C_i^t, i \notin \{j, k\} : w'_i(j, k) = \min\{w_i^j(j, k), w_i^k(j, k)\} \quad (7)$$

Let $MF(i, j)$ denote the maximum flow from i to j in G'_i . The mechanism sets the reputation of agent j as follows: $R_i(j) = MF(j, i) - MF(i, j)$.

In the Drop-Edge Mechanism, the behavior is the same as BarterCast when agent i has private information about an edge. Otherwise, lines (4)-(6) implement the "edge-dropping" idea. Any report from an agent in the choice set is dropped and an edge (j, k) is dropped completely if both j and k are in the choice set. Finally, when two reports are available the information is combined via a min operator just as for BarterCast. Note that we could use max without compromising the incentive properties of our mechanism, however we use min to simplify the comparison with BarterCast for now. In general, using max would promote improved informativeness by taking the more recent report (since reports of work only increase.)

PROPOSITION 1. The Drop-Edge Mechanism is misreportproof.

PROOF. No report of an agent i is used in making a decision whenever agent i is in the choice set of any other agent j . \square

Note that we do not need the use of maxflow for the mechanism to be misreportproof. On the other hand, this is retained to allow for a direct comparison with BarterCast and because it seems to have an essential role in protecting against sybil attacks while still providing some degree of informativeness.⁶

5. EXPERIMENTAL EVALUATION

We have shown in the last section that the Drop-Edge mechanism is misreportproof, and thus has better robustness against manipulation than BarterCast. However, this does not tell us about its usefulness in practice. We are dropping edges to achieve this additional robustness which implies that some information will be lost. In this section, we evaluate the two mechanisms empirically via simulation to better understand the trade-off between informativeness and incentive compatibility that the mechanisms are making.

5.1 Experimental Set-up

We simulate a P2P file-sharing environment with 100 agents and discrete time steps. Downloading a file is consuming work and uploading a file is performing work. In every time step, every agent decides whether to upload one unit of work to the other agents in the network or not. Agents are divided into cooperative and malicious agents— cooperative agents always upload one unit of work, while malicious agents only upload in every other round. A fraction β of the agents are malicious. Furthermore, we also have a fraction γ of all agents that are strategic: these agents are malicious but in addition seek to manipulate the accounting mechanism

⁶Without this concern, though, one could in fact define a variation of the mechanism that simply computes the "perfect" net contribution of agent $j \in C_i^t$ given modified, subjective work graph G'_i , as $R_i(j) = \sum_{k|(j,k) \in E'_i} w'_i(j, k) - \sum_{k|(k,j) \in E'_i} w'_i(k, j)$. This would remain misreportproof.

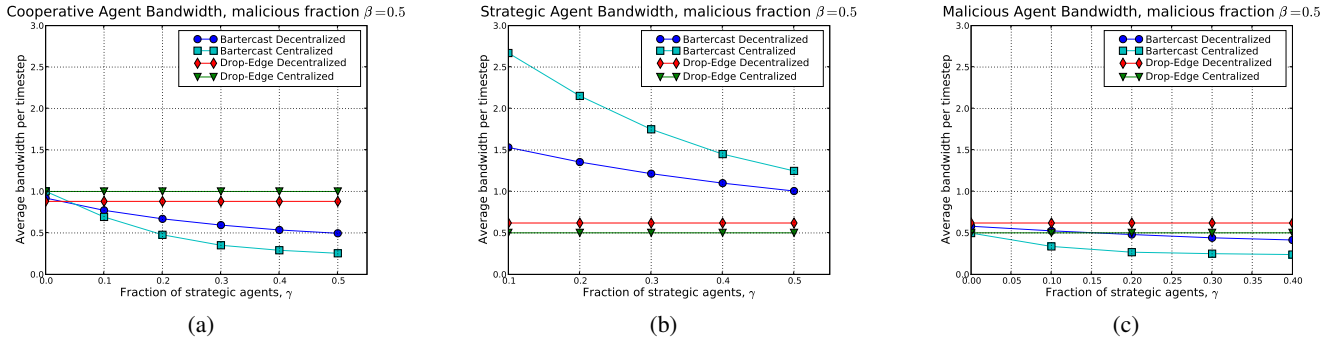


Figure 4: Bandwidth achieved by different groups of agents under the different accounting mechanisms.

through misreports when this is advantageous (i.e. when they can increase their reputation or work consumed).

In every round, when an agent decides to do work for the system, he gets a random choice set of 5 agents. With probability 0.1 he uploads 1 unit to a random agent (simulating optimistic unchoking), and with probability 0.9, he uses his accounting mechanism and the allocation rule to determine whom to upload to. After every round, every agent contacts one other agent at random to exchange messages about their direct experiences in the network. All agents send a report about the last 5 agents they have interacted with and about the 5 agents that have uploaded the most to them. Both cooperative and malicious, non-strategic agents will make these reports truthfully. Strategic agents perform misreport manipulations.

For both the Bartercast and Drop-Edges mechanisms we run a centralized and decentralized version for the experiment. The centralized version proceeds like the decentralized version, except no message passing takes place. Reports are made directly to a central entity, and every agent has access to those reports. Thus, agents can still misreport their true values to the mechanism, while maintaining true information in their local graphs. Note that the centralized Drop-Edge mechanism also drops the edges from the choice sets. Also considering the centralized versions of the mechanisms helps isolate the effect of the message passing algorithm.

We run each simulation for 100 time steps, recording the amount each agent has uploaded and downloaded. This allows us to determine absolute and relative efficiency statistics about the different agent groups. We measure the average amount of bandwidth received by each type of agent per time step. Each experiment is averaged over 20 trial runs.

5.2 Informativeness & Efficiency Results

Informativeness: The first thing we want to check is the informativeness of the two mechanisms. We measure this indirectly by considering the efficiency measurements when no strategic agents are present. We look at the graphs in Figure 4 (a) with zero strategic agents, i.e. where $\gamma = 0$. We expect Drop-Edge to be slightly less informative because we are dropping information that BarterCast is using. And in fact we see that the efficiency is higher under both the centralized and decentralized versions of the Bartercast mechanism, but only minimally so (and only with zero strategic agents). Thus, the solution to the misreport manipulations of BarterCast make the Drop-Edge mechanism only slightly less informative in the absence of strategic agents.

Efficiency: The more interesting experimental analysis concerns the overall efficiency of the system with malicious and strategic agents. *It is our goal to maximize the efficiency of the cooperative agents and to minimize the efficiency for malicious agents, and for strategic agents in particular.* In the P2P filesharing setting, the efficiency is measured as the amount of bandwidth downloaded/consumed: we would like to reward peers who share more with more download bandwidth.

By comparing Figures 4 (b) and (c), we see the relative efficiency of the strategic agents compared to the malicious agents under the Bartercast and Drop-Edges mechanisms. It is important to note that strategic agents are able to sharply increase their average performance by misreporting under the Bartercast mechanism. The bandwidth received by strategic agents is significantly higher than that of the good and malicious agents. This effect is particularly high when only a few strategic agents are in the system. With 10% strategic agents, the performance of a strategic agent is 3 times as high as that of the other agents under the decentralized BarterCast mechanism, and more than 5 times as high under the centralized BarterCast mechanism. Thus, with BarterCast, agents have a very large incentive to act strategically. The Drop-Edge mechanism in contrast leads to the same constant efficiency for strategic as for malicious agents, and the performance of cooperative agents is almost twice as high as that of malicious and strategic agents.

Efficiency over time: We also ran a longer experiment with $\beta = 0.5$, $\gamma = 0.2$ for 500 trials, measuring efficiency over the course of the simulation. In Figure 5, we see that the performance improvements that strategic agents gain from misreporting get even larger over time. The misreport attack is effective at increasing their reputations and increasing the amount of download bandwidth they receive. This occurs at the expense of the other agents in the system. Compare this against Figure 6, which presents results for the same simulation under the Drop-Edge mechanism. Strategic agents cannot manipulate their reputations, and receive decreasing amounts of bandwidth as the simulation proceeds. At the end of the run, good agents receive more than twice as much bandwidth per round.

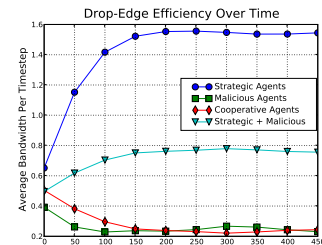


Figure 5: Bandwidth in BarterCast Mechanism over Time.

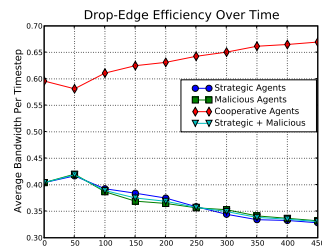


Figure 6: Bandwidth in Drop-Edge Mechanism over Time.

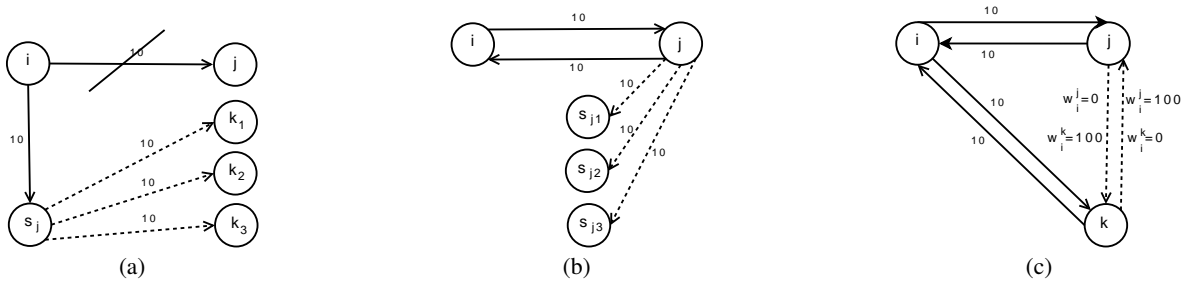


Figure 7: (a) A sybil attack where agent j uses his sybil s_j to decrease the reputation of the agents $k_1, k_2,$ and k_3 . (b) A sybil attack where agent j generates many sybils, then does a little bit of work for i and then provides its sybils with positive reputation. (c) A collusion attack involving agents j and k which only works for the Drop-Edge mechanism. Dotted edges indicate false reports.

6. SYBIL ATTACKS AND COLLUSION

We begin this section addressing the well-known problem of sybil attacks. However, as we have pointed out earlier, sybil attacks in distributed work systems are much more powerful than in the standard transitive trust scenarios. For example, the max-flow reputation mechanism has been shown to be sybil-proof [4], but this is only for the weaker notion that an agent cannot increase *its own* reputation by using a sybil attack. In our environment, both the BarterCast and the Drop-Edge mechanisms are susceptible to sybil attacks despite using variants of the max-flow mechanism.

Consider graph (a) in Figure 7 where we illustrate the first sybil attack. Here, agent j , instead of receiving directly 10 units from i , generates sybil s_j and lets the sybil receive 10 units from i . That sybil can now claim that it has done 10 units of work for all other agents $k_1, k_2,$ and k_3 . Thus, after the attack, agent j has a reputation value of 0, while the agents $k_1, k_2,$ and k_3 all have a reputation of -10. This attack was possible because s_j had received 10 units of work from i and now i allows s_j to make arbitrary reports about other agents up to 10 units of work. But why should agent i trust agent j 's reports in the first place?

It turns out that this is exactly the crux of the problem with these two mechanisms in regard to sybil attacks. By using the max-flow mechanism in this way, the semantics of transitive-trust mechanisms unravels. The general idea of transitive trust is "I trust you and your reports if I know you are good." However, in this sybil attack, agent i trusted the reports of s_j although s_j never did any work for i , i.e., never proved to be trustworthy. A solution that would fix this sybil attack would be an *asymmetric max-flow mechanism* that only propagates flow from any node k (that makes a report) to i along paths directed towards i , because all of the edges on that path correspond to work and thus provide some evidence about the trustworthiness of the corresponding agents. We are currently analyzing the resulting asymmetric accounting mechanisms analytically and experimentally.

Now, consider graph (b) in Figure 7 where we illustrate the second sybil attack. Here, agent j does a little bit of work for agent i and then propagates the resulting trustworthiness to all of its sybils, providing them with positive reputation. This sybil attack is particularly powerful because as a result of the manipulation, all of j 's sybils can now receive work from i , and in general, j could just keep producing new sybils ad infinitum. The problem with this attack is that if agent j has earned some trustworthiness *once*, j can benefit from that infinitely often. A solution to this problem seems very challenging. One idea is to also track the total amount of flow on an edge (i, j) that has been used by any agent in determining the reputation assigned to some other agent k that is the selected to receive work. The amount of flow that is tracked as having been "utilized" is limited to be no more than the actual amount of work performed by j and thus j 's influence (on its sybils or otherwise) is bounded. In a decentralized system, correctly keeping track of these utilizations would require an additional message protocol that distributes this information. We are currently analyzing this ap-

proach analytically and experimentally. Ultimately, it is our goal to design accounting mechanisms that are *strongly sybil-proof*, i.e., that are robust to all of these sybil manipulations.

Finally, we want to briefly point out an interesting collusion attack that is only possible with the Edge-Drop mechanism. Consider graph (c) in Figure 7 where agents j and k collude. Both agents make two reports: first, they report that the other agent has done a lot of work for them and second they report that they have not done any work for the other agent. This collusion works because both agents know that whenever they are in a choice set, their report will be dropped anyways, and thus, they are not suffering from their own negative report about themselves, but the other agent might benefit from the positive report about him. However, this collusion does not seem to be a problem in practice because it would be very difficult to implement. Agent j only gains any benefit if agent k adheres to the collusion contract, but this is hard to verify for agent j . Thus, there is neither an incentive on k 's side nor on j 's side to change/hack the protocol (assuming that changing the protocol has some small ε cost) and the other agent will never find out. Thus, if agents j and k are rational, then in equilibrium they both won't manipulate. This is a nice example where the traditional "tragedy of the commons" actually leads to a positive overall outcome for the rest of the community.

7. CONCLUSION AND FUTURE WORK

In this paper we have formalized the problem of designing incentive-compatible accounting mechanisms for distributed work systems and modified BarterCast to create Drop-Edge, which is robust to misreport attacks. In future work, we will continue the analysis of sybil attacks on accounting mechanisms, with the ultimate goal of a *strong* sybilproof accounting mechanism. Because a strong sybilproof mechanism may turn out not to be very informative, we will then make use of more simulation experiments to find a good trade-off between sybilproofness and informativeness. We conjecture that it is very hard to simultaneously prevent attacks of both 3rd and 4th order in the same accounting mechanism without losing any meaningful informativeness, and expect that the next advance will come by preventing attacks of 3rd order and leveraging a problem asymmetry in the willingness to prevent 3rd order while allowing 4th order attacks.

Acknowledgements

We are thankful to Michel Meulpolder and Johan Pouwelse for helpful discussions on this work. The first author is supported in part by a Microsoft Research Fellowship.

8. REFERENCES

- [1] A. Altman and M. Tennenholtz. On the Axiomatic Foundations of Ranking Systems. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 917–922, 2005.

- [2] A. Altman and M. Tennenholtz. An Axiomatic Approach to Personalized Ranking Systems. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007.
- [3] Y. Benkler. Coase's Penguin, or, Linux and The Nature of the Firm. *The Yale Law Journal*, 112(3):369–446, 2002.
- [4] A. Cheng and E. Friedman. Sybilproof Reputation Mechanisms. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, pages 128–132, Philadelphia, PA, August 2005.
- [5] A. Cheng and E. Friedman. Manipulability of PageRank under Sybil Strategies. In *Proceedings of the First Workshop of Networked Systems (NetEcon06)*, 2006.
- [6] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, Berkeley, CA, June 2003.
- [7] E. J. Friedman, J. Y. Halpern, and I. Kash. Efficiency and Nash Equilibria in a Scrip System for P2P Networks. In *Proceedings of the 7th ACM Conference on Electronic Commerce (EC)*, pages 140–149, Ann Arbor, Michigan, 2006.
- [8] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Twelfth International World Wide Web Conference*, pages 640–651, 2003.
- [9] I. A. Kash, E. J. Friedman, and J. Y. Halpern. Manipulating scrip systems: Sybils and collusion. In *Proceedings of the First Conference on Auctions, Market Mechanisms and Their Applications (AMMA)*, Boston, MA, May 2009.
- [10] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips. Bartercast: A practical approach to prevent lazy freeriding in p2p networks. In *Proceedings of the 6th International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P)*, Rome, Italy, May 2009.
- [11] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 1–14, San Francisco, California, April 2008.
- [12] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-Sharing System: Measurements and Analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.
- [13] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steelen, and H. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 19, 2008.
- [14] D. Sheldon and J. Hopcroft. Manipulation-Resistant Reputations Using Hitting Time. In *5th Workshop on Algorithms for the Web-Graph*, 2007.