Accounting Mechanisms for Distributed Work Systems

Sven Seuken
School of Engineering & Applied Sciences
Harvard University
Cambridge, MA 02138
seuken@eecs.harvard.edu

Jie Tang
EECS Department
University of California, Berkeley
Berkeley, CA 94720
jietang@eecs.berkeley.edu

David C. Parkes
School of Engineering & Applied Sciences
Harvard University
Cambridge, MA 02138
parkes@eecs.harvard.edu

Abstract

In distributed work systems, individual users perform work for other users. A significant challenge in these systems is to provide proper incentives for users to contribute as much work as they consume, even when monitoring is not possible. We formalize the problem of designing *incentive-compatible accounting mechanisms* that measure the net contributions of users, despite relying on voluntary reports. We introduce the Drop-Edge Mechanism that removes any incentive for a user to manipulate via misreports about work contributed or consumed. We prove that Drop-Edge provides a good approximation to a user's net contribution, and is accurate in the limit as the number of users grows. We demonstrate very good welfare properties in simulation compared to an existing, manipulable mechanism. In closing, we discuss our ongoing work, including a real-world implementation and evaluation of the Drop-Edge Mechanism in a BitTorrent client.

1 Introduction

Distributed work systems arise in many places where individual users perform work for other users, often called peer production. For example, users of a peer-to-peer (P2P) file-sharing network share videos, music or software with each other. Amazon Mechanical Turk and other "crowd sourcing" applications suggest an explosion of interest in new paradigms for economic production. Within AI, distributed work systems contribute to the agenda on multi-agent resource allocation, enhancing our understanding of architectures to coordinate artificial agents.

Of course, the total work performed by a 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 transfers must rely on roughly balanced work contributions. Often time this is achieved by seeking to enforce temporally-local balance, e.g., via fair exchange protocols such as BitTorrent. 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 consumption (Piatek et al. 2008).[1]

The problem of designing accounting mechanisms is motivated by this need to tally work contributed and consumed in order to improve system efficiency by temporally decoupling these activities. The particular challenge that we address here occurs when there is no ability for a third party to monitor the activities, and instead tallies rely on voluntary reports by the same users that may seek to free-ride. Thus, an accounting system must be robust to manipulations, because users might overstate the amount of work contributed, or understate the amount of work consumed.

Accounting vs. Reputation Mechanisms. The problem of designing an incentive-compatible accounting mechanism shares some features with work on trust/reputation mechanisms (Friedman, Resnick, and Sami 2007). However, there are significant differences which make reputation and accounting mechanisms incomparable. First, and somewhat informally, the essence of accurate reputation aggregation is the operation of *averaging* whereas the essence of accurate accounting is the operation of *addition*. In a reputation system like eBay, individual users provide feedback about each other, and the individual feedback reports of two different agents regarding a third agent could be very different. The task of the reputation system is to aggregate multiple reports into one overall reputation score; in a sense, "averaging" over all reports. In contrast, in distributed work systems, multiple reports about work consumed or performed by an agent simply need to be added together, to eventually determine the overall net contributions of that agent.

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, mechanisms that are sybilproof in traditional reputation systems are not necessarily sybilproof in distributed work systems.

[1](Pouwelse et al. 2005) found that more than 80% of BitTorrent users go offline immediately once they have finished downloading. Accounting mechanisms would solve this problem by giving users an incentive to share even after they have finished downloading.

Overview of Results. We present what we believe to be the first formal model for the design of incentive-compatible accounting mechanisms for distributed work systems. We suggest a simple misreport attack on *BarterCast*, an existing accounting mechanism that is already implemented in the BitTorrent client *Tribler* (Meulpolder et al. 2009), and introduce the *Drop-Edge Mechanism*, a new mechanism that is robust to misreports. We show analytically that Drop-Edge provides a good approximation to a user's net contribution, and is accurate in the limit as the number of users grows. Furthermore, our simulation results suggest that the effect of misreports in BarterCast can significantly reduce social welfare, while Drop-Edge achieves much better efficiency.

Related Work. Our work is motivated by BarterCast (Meulpolder et al. 2009), a decentralized accounting mechanism that is effective in distinguishing between cooperative agents and free-riders. However, the authors do not analyze the attacks presented here in detail. (Piatek et al. 2008) also study decentralized accounting and find empirically that most users of P2P file-sharing networks are connected via a one hop link in the connection graph. They propose to use well-connected intermediaries to broker information, but without providing proper incentives to the intermediaries to behave truthfully. Feldman et al. (2004) study the challenges in providing robust incentives in fully decentralized P2P networks, including potential misreport attacks. However, they do not propose a misreport-proof mechanism. Interestingly, our mechanism shares some similarities with a mechanism proposed by Alon et al. (2010), who consider voting environments where the set of candidates coincides with the set of voters. However, they don't consider the problem of transitive trust in networks. A recent paper by Resnick and Sami (2009) addresses this exact problem in detail, and they show novel results about sybilproof trust mechanisms. In their domain they do not consider shared trust information and thus they do not have to study misreport attacks. An interesting, but orthogonal direction, is provided by studies of virtual currency systems (Friedman, Halpern, and Kash 2006). There, work provision or consumption is observable and there is a trusted currency.

2 Formal Model

Consider a distributed work system of n agents each capable of doing work for each other. All work is assumed to be quantifiable in the same units. The work performed by all agents is captured by a work graph:

Definition 1. ***(Work Graph)*** *A work graph $G = (V, E, w)$ has vertices $V = \{1, \ldots, n\}$, one for each agent, and directed edges $(i, j) \in E$, for $i, j \in V$, 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 true work graph is unknown to individual agents because agents only have direct information about their 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 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.*

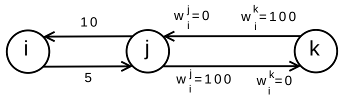


Figure 1: A subjective work graph from agent i's perspective. Edges where i has direct information have only one weight. Other edges can have two weights, corresponding to the possibly conflicting reports of the two agents involved.

Based on its own experiences and known reports from other agents, agent i can construct a subjective work graph (see Figure 1). Let $w_i^j(j, k), w_i^k(j, k) \in \mathbb{R}_{\geq 0}$ denote the edge weight as reported by agent j and agent k respectively.

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 $w_i^j(j, k), w_i^k(j, k)$ as known to i. 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 the edge weights $w_i^j(j, k)$ and $w_i^k(j, k)$ need not be truthful reports about $w(j, k)$. These weights may propagate through a centralized mechanism, where a center shares received reports with all agents, or a decentralized information exchange protocol, where agents exchange information about work performed and received bilaterally.

Periodically, an agent i can receive a work request by a set of agents with which the agent may have rarely or never interacted with before. This induces a choice set:

Definition 4. ***(Choice Set)*** *We let $C_i \subseteq V \setminus \{i\}$ denote the choice set for agent i, i.e., the set of agents that are currently interested in receiving some work from i.*

We assume that an agent has no *a priori* bias towards assisting one agent over another. The role of an accounting mechanism is to provide an estimate of the net work contributed by each agent $j \in C_i$ to the system.

Definition 5. ***(Accounting Mechanism)*** *An accounting mechanism M takes as input a subjective work graph G_i, a choice set C_i, and determines the score $S_j^M(G_i, C_i)$, for any agent $j \in C_i$, as viewed by agent i.*[2]

Based on the results of the accounting mechanism, the agent then decides who to allocate work to. We use the following allocation method in this paper:

Definition 6. ***(Winner-takes-all Allocation)*** *Given subjective work graph G_i, choice set C_i, and accounting mechanism M, agent i performs one unit of work for agent $j \in \arg\max_{k \in C_i} S_k^M(G_i, C_i)$, breaking ties at random.*

In general, other allocation rules can be used, e.g., proportional allocation, or threshold rules.

[2]Note that we purposefully chose to use the term "score" instead of "reputation value" even though this is in contrast to prior work by Meulpolder et al. (2009). Our goal is to clearly distinguish between accounting and reputation mechanisms and to emphasize that outputs of such mechanisms have very different meanings.

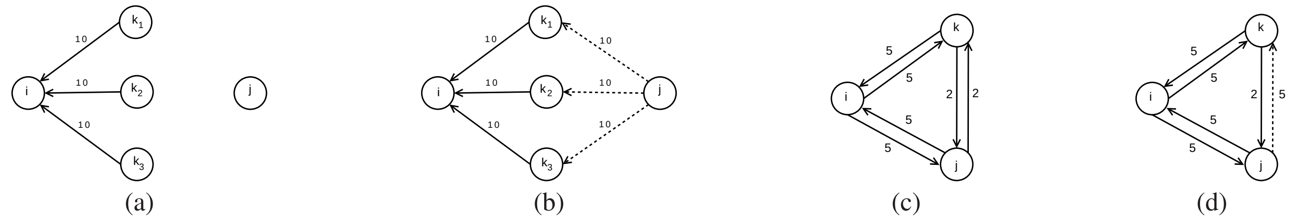


Figure 2: (a) Work graph 1 based on true reports. (b) Subjective work graph as viewed by i, including a misreport attack by j to boost its score in BarterCast. (c) Work graph 2 based on true reports. (d) Subjective work graph 2 as viewed by i, including a misreport attack by j to decrease k's score and increase its own score. Dotted edges indicate misreports.

Strategic Manipulations

We adopt the model and terminology of Meulpolder et al. (2009), and assume a population that consists of a mixture of *cooperative* agents (or sharers), who always contribute work, and *lazy free-riders* who intermittently shirk work. The role of an accounting mechanism is to make it unbeneficial to be a free-rider. We further model a subset of the free-riding agents as *strategic* agents, who also try to manipulate the accounting mechanism itself through misreport attacks, where an agent reports false information about its work performed or consumed. The non-strategic free-riders are called "lazy" because they try to avoid performing work, but they are too lazy to perform any kind of manipulations.[3]

Definition 7. ***(Misreport-proof)*** *An accounting mechanism M is misreport-proof if, for any agent $i \in V$, any subjective work graph G_i, any choice set C_i, any agent $j \in C_i$, for every misreport manipulation by j, where G'_i is the subjective work graph induced by the misreports, the following holds:*

- $S_j^M(G'_i, C_i) \leq S_j^M(G_i, C_i)$, *and*
- $S_k^M(G'_i, C_i) \geq S_k^M(G_i, C_i)\ \forall k \in C_i \setminus \{j\}$.[4]

Given a misreport-proof accounting mechanism, we assume that strategic agents will choose to follow the protocol and be truthful. We assume that the accounting mechanism itself is fixed along with the allocation rule of an agent. We model only strategic behavior with regard to reports about work performed or consumed. The motivating assumption is that agents are *a priori* indifferent as to whom they work for, with free-riders having preferences only in favor of receiving work and against performing work.

3 The Drop-Edge Mechanism

In this section we first review BarterCast, the mechanism introduced by Meulpolder et al. (2009). We then demonstrate two misreport attacks on BarterCast. Finally, we introduce our Drop-Edge Mechanism which is misreport-proof.

[3]Another class of attacks are sybil attacks, where an agent introduces sybil nodes (fake agents) that perform and consume work. In practice, misreport attacks are much easier to execute than sybil attacks and also more beneficial. In contrast, sybil attacks might require adopting multiple IP addresses and coordinating work between them. According to Pouwelse et al., they have not yet seen sybil attacks in practice, while misreport attacks have a long history (c.f. Kazaa Lite, eDonkey). Thus, for practitioners, protecting against misreport attacks is of higher importance.

[4]Note that the first requirement is equivalent to *value-strategyproofness* as defined for trust mechanisms, and both requirements together imply *rank-strategyproofness* (Cheng and Friedman 2006).

Definition 8. ***(BarterCast Mechanism)*** *Given subjective work graph G_i and choice set C_i, construct a modified graph $G_i^B = (V_i, E_i, w_i^B)$ with weights defined as:*

$$\forall (j,k) | i \in \{j,k\} : w_i^B(j,k) = w_i^i(j,k)$$

$$\forall (j,k) | i \notin \{j,k\} : w_i^B(j,k) = \max\{w_i^j(j,k), w_i^k(j,k)\},$$

where missing reports in the max-operator are set equal to 0. Let $MF_{G_i^B}(i,j)$ denote the maximum flow from i to j in G_i^B. Define the BarterCast Score of agent j as $S_j^B(G_i, C_i) = MF_{G_i^B}(j,i) - MF_{G_i^B}(i,j)$.[5]

In BarterCast, an agent takes its own information over reports from others. Given two reports, it takes the maximum of the two. Note that even if no agents misreport, two reports for the same edge will generally be in conflict when a decentralized mechanism is being used due to the decentralized information exchange protocol. By taking the maximum of the two reports, an agent always uses the most up-to-date information it has. The max-flow algorithm bounds the influence of any report that agent j can make by the edges between i and j, preventing an agent from grossly inflating the work it has performed for another agent. This limits the power of strategic manipulations and also protects against Byzantine attacks (i.e., arbitrary attacks not necessarily originating from rational agents). In some sense, using max-flow can be seen as performing a form of *bounded addition*.

BarterCast can be manipulated via misreports. In fact, *it is a dominant strategy in BarterCast to always report ∞ work performed, and 0 work consumed.* We illustrate two attacks in Figure 2. We always show the subjective work graph from i's perspective and the manipulating agent is j. Graph (a) shows a true work graph. Graph (b) shows agent i's view of the work graph, now including a misreport by agent j. Agent j has simply reported that it has done work for k_1, k_2, and k_3, although it did not. BarterCast cannot catch this: because there never was an interaction there are no reports from these other agents. Note that agent j increased its score from 0 to 30. Now consider Graph (c) which shows a new true work graph. Graph (d) shows a misreport manipulation by agent j where j reported that it has done 5 units of work for k even though it only did 2 units of work. Because BarterCast takes the maximum of two reports, agent i will believe j. As a result, agent k's score has decreased from 0 to -3, and agent j's score has increased from 0 to 3.

[5]This specification of BarterCast differs from Meulpolder et al. (2009) only in that they take the arctan of the difference between the flows. However, because arctan is a monotonic function this does not change the ranking of the agents.

We now introduce a powerful alternative to BarterCast:

Definition 9. ***(Drop-Edge Mechanism)*** *Given subjective work graph G_i and choice set C_i, construct the modified graph $G_i^D = (V_i, E_i, w_i^D)$ with the weights w_i^D defined as:*

$$\forall(j,k)|i \in \{j,k\} : w_i^D(j,k) = w_i^i(j,k)$$

$$\forall(j,k)|j,k \in C_i : w_i^D(j,k) = 0 \quad (1)$$

$$\forall(j,k)|j \in C_i, k \notin C_i : w_i^D(j,k) = w_i^k(j,k) \quad (2)$$

$$\forall(j,k)|k \in C_i, j \notin C_i : w_i^D(j,k) = w_i^j(j,k) \quad (3)$$

$$\forall(j,k)|j,k \notin C_i, i \notin \{j,k\} : w_i^D(j,k) = \max\{w_i^j(j,k), w_i^k(j,k)\}.$$

Missing reports in the max-operator are set to 0. *Agent j's score is $S_j^D(G_i,C_i) = MF_{G_i^D}(j,i) - MF_{G_i^D}(i,j)$.* [6]

Lines (1)-(3) implement a simple "edge-dropping" idea. Any reports received by agent i from agents in the choice set C_i are dropped in determining edge weights in modified graph G_i^D. An edge (j,k) is dropped completely if both j and k are inside C_i. We make the following observation:

Proposition 1. *Drop-Edge is misreport-proof.*

Proof. No report of agent j is used in i's decision making process whenever agent j is in the choice set of agent i. □

4 Theoretical Analysis

In this section we analyze the information loss of Drop-Edge due to the discarded edges. All statements are w.r.t. the centralized version of Drop-Edge only. The analysis is based on agent i's subjective work graph $G_i = (V_i, E_i, w_i)$. $G_i^D = (V_i, E_i, w_i^D)$ denotes the modified graph after the Drop-Edge mechanism has been applied to G_i, and $G_i^O = (V_i, E_i, w_i^O)$ analogously for the omniscient mechanism (which adjusts weights like BarterCast would). For both Drop-Edge and the omniscient mechanism, we assume that agents do not perform manipulations. For the first theorem, we only consider the information loss in the work graph that Drop-Edge produces. We later add the use of max-flow to the analysis. For graph $G_i = (V_i, E_i, w_i)$, we define the net work on edge (k,j) as $\tilde{w}_i(k,j) = w_i(k,j) - w_i(j,k)$ so that the overall net work is $work_i(k, G_i) = \sum_{j \neq k} \tilde{w}_i(k,j)$.

Theorem 1. *For all subjective work graphs $G_i = (V_i, E_i, w_i)$ with $|V_i| = n$, for all $k \in V_i$, for all choice sets C chosen uniformly at random with $|C| = m$ and $k \in C$:*

$$\frac{\mathbb{E}_C[work_i(k, G_i^D)]}{work_i(k, G_i^O)} = 1 - \frac{(m-1)}{(n-1)}.$$

Proof.

$$\mathbb{E}_C[work_i(k, G_i^D)] = \mathbb{E}_C[\sum_{j \neq k} \tilde{w}_i^D(k,j)] = \sum_{j \neq k} \mathbb{E}_C[\tilde{w}_i^D(k,j)]$$

$$= \sum_{j \neq k} \Big[\frac{(m-1)}{(n-1)} \cdot 0 + \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot \tilde{w}_i^O(k,j)\Big] \quad (4)$$

$$= \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot work_i(k, G_i^O)$$

For equation 4, consider edge (k,j). Because C is chosen uniformly at random with $k \in C$, the probability that j is also inside any random C is $\frac{m-1}{n-1}$. If k and j are inside C the edge gets dropped, otherwise $\tilde{w}_i^O(k,j)$ is counted. □

[6]Note that we do not need max-flow for misreport-proofness. However, we retain it because it makes a more direct comparison with BarterCast possible and it protects against Byzantine attacks.

Theorem 1 implies that if n is relatively large compared to m, then the expected net work computed by the Drop-Edge Mechanism is very close to the true net work.[7] The following corollary states this nice property more formally:

Corollary 1. *For all subjective work graphs $G_i = (V_i, E_i, w_i)$ with $|V_i| = n$, for all $k \in V$, for choice sets C chosen uniformly at random with $|C| = m$, it holds that:*

$$\lim_{\frac{n}{m} \to \infty} \frac{\mathbb{E}_C[work_i(k, G_i^D)]}{work_i(k, G_i^O)} = 1.$$

We now turn our attention to the approximation ratio of the scores computed by Drop-Edge when the max-flow algorithm is used. We consider max-flows restricted to a certain number of hops, and let $MF_{G,h}(i,j)$ denote the max-flow from node i to j in graph G with exactly h hops. We let $S_{j,h}^D(G_i, C)$ denote the score computed by Drop-Edge for h hops, i.e., $S_{j,h}^D(G_i, C) = MF_{G_i^D,h}(j,i) - MF_{G_i^D,h}(i,j)$. Analogously, $S_{j,h}^{=}(G_i, C)$ is the score computed by an omniscient mechanism for exactly h hops.

Theorem 2. *For all subjective work graphs $G_i = (V_i, E_i, w_i)$ with $|V_i| = n$, for all $k \in V_i$, for i's choice set $C_i = C$ chosen uniformly at random with $|C| = m$ and $k \in C$:*

$$E_C[S_k^D(G_i, C)]$$

$$= E_C[S_{k,0}^O(G_i, C)] + \sum_{h=1}^{n-m-1} \prod_{p=1}^{h} \big(\frac{n-m-p}{n-1-p}\big) \cdot E_C[S_{k,h}^O(G_i, C)].$$

Proof.

$$E_C[S_k^D(G_i, C)] = E_C[MF_{G_i^D}(k,i) - MF_{G_i^D}(i,k)] \quad (5)$$

$$= \sum_{h=0}^{n-m-1} E_C[MF_{G_i^D,h}(k,i) - MF_{G_i^D,h}(i,k)] \quad (6)$$

$$= E_C[MF_{G_i^D,0}(k,i) - MF_{G_i^D,0}(i,k)] \quad (7)$$

$$+ \sum_{h=1}^{n-m-1} E[MF_{G_i^D,h}(k,i) - MF_{G_i^D,h}(i,k)] \quad (8)$$

$$= E_C[S_{k,0}^O(G_i, C)] \quad (9)$$

$$+ E_C[MF_{G_i^D,1}(k,i) - MF_{G_i^D,1}(i,k)] \quad (10)$$

$$+ \sum_{h=2}^{n-m-1} E_C[MF_{G_i^D,h}(k,i) - MF_{G_i^D,h}(i,k)] \quad (11)$$

$$= E_C[S_{k,0}^O(G_i, C)] + \big(\frac{n-m-1}{n-2}\big) E_C[S_{k,1}^O(G_i, C)] \quad (12)$$

$$+ \big(\frac{n-m-1}{n-2}\big) \cdot \big(\frac{n-m-2}{n-3}\big) E_C[S_{k,2}^O(G_i, C)] \quad (13)$$

$$+ \sum_{h=3}^{n-m-1} E_C[MF_{G_i^D,h}(k,i) - MF_{G_i^D,h}(i,k)] \quad (14)$$

$$= E_C[S_{k,0}^O(G_i, C)] + \sum_{h=1}^{n-m-1} \prod_{p=1}^{h} \big(\frac{n-m-p}{n-1-p}\big) \cdot E_C[S_{k,h}^O(G_i, C)]$$

In Equation 7 we isolated the expectation of the 0-hop max-flow terms, which are direct paths between i and k and thus

[7]Note that Theorem 1 holds for any graph, including power-law graphs which we would expect in the file-sharing domain. There, a choice set size of 50 and a graph size larger than 100,000 are reasonable. The expected ratio would then already be above 0.9995.

the edges are not dropped and Equation 9 follows. In Equation 10 we isolated the expectation of the 1-hop max-flow terms which are paths of length 2 between i and k. Because C was chosen uniformly at random, the probability that the intermediate node lies outside of C is $\frac{n-m-1}{n-2}$ and Equation 12 follows. The final expression follows from analogous reasoning for all h-hop max-flows. □

In practice, running the max-flow algorithm on the full work graph takes too long. The following corollary tells us the accounting accuracy for a restricted max-flow algorithm:

Corollary 2. *Let H denote the max number of hops used in computing max-flow. Then, for all subjective work graphs $G_i = (V_i, E_i, w_i)$ with $|V_i| = n$, for all $k \in V_i$, for i's choice set $C_i = C$ chosen uniformly at random with $|C| = m$ and $k \in C$:*

$$E_C[S^D_{k,H}(G_i, C)] \geq \left(\frac{n-m-H}{n-H}\right)^H E_C[S^O_{k,H}(G_i, C)].$$

In the BarterCast implementation (Meulpolder et al. 2009) and also in our experiments, the max-flow is restricted to at most 1 hop (i.e., paths of length at most 2). For that particular mechanism we get:

$$\frac{E_C[S^D_{i,1}(G_i, C)]}{E_C[S^O_{i,1}(G_i, C)]} \geq (\frac{n-m-1}{n-2}).$$

Note that the theoretical results bound the accuracy in expectation over choice sets and don't directly pertain to accuracy with respect to selecting the right agent from a given choice set. Fortunately, the experimental results we present in the next section show that Drop-Edge indeed performs very well in practice.

5 Experimental Evaluation

In this section, we evaluate the mechanisms empirically via simulation to better understand the trade-offs that are made in the Drop-Edge and BarterCast mechanisms.

Experimental Set-up. We simulate a P2P file-sharing environment with 100 agents and discrete time steps. Downloading a file corresponds to consuming work and uploading a file corresponds to performing work. In every time step, every agent decides whether to perform one unit of work or not. Agents are divided into a fraction $1-\beta$ of cooperative and a fraction β of free-riding agents. Cooperative agents always perform one unit of work, while free-riders only perform work in every other round. Furthermore, we also model strategic free-riding agents who seek to manipulate the accounting mechanism. We let $\gamma \leq \beta$ denote the total fraction of all agents that are strategic free-riders. With BarterCast, the strategic agents perform the optimal misreport manipulation, i.e., always reporting they have consumed 0 units of work and contributed ∞ units of work.

In each round that agent i performs work, it gets a random choice set of 5 agents. With probability 0.1, i performs 1 unit of work for a random agent in the choice set, and with probability 0.9 it uses the accounting mechanism and allocation rule to determine who receives work. This simulates the "optimistic unchoking" found to be useful in distributed work systems such as BitTorrent. Each 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 the 5 agents that have uploaded the most to them. Strategic agents are untruthful when sending these reports.

For both Bartercast and Drop-Edge we run a centralized and decentralized version for the experiment. In the centralized version, all reports (which can still be untruthful) are made directly to a central entity, immediately after each interaction, and are then available to every agent. Considering the centralized version of each mechanism helps isolate the effect of the message passing algorithm. We run each simulation for 100 time steps and record the work contributions and consumptions (averaged over 10 trials).

Results. We first verify our theoretical results on information loss from Section 4. Fixing a choice set size $m = 5$, free-rider agent fraction $\beta = 0.5$, and strategic agent fraction $\gamma = 0.2$, we simulate networks of size $n = 10, 20, ...200$. After 100 time steps, for every agent we randomly choose a choice set and measure for every agent in the choice set the ratio of the Drop-Edge score and the score under the omniscient mechanism. Averaging over all agents and choice sets, we find that our empirical results closely match the theoretical results from Corollary 2 (Figure 3).

Now that we have established experimentally that at least on average, Drop-Edge provides a good approximation to the agents' scores, we turn our attention to directly measuring the mechanisms' performance. First, we measure their performance without strategic agents, to isolate their effectiveness as algorithms in aggregating information and promoting good decisions. Consider the graphs in Figure 4 (a) with zero strategic agents, i.e. where $\gamma = 0$. We expect Drop-Edge to be slightly less efficient because we are dropping information that BarterCast is using, and no strategic agents are present that could harm BarterCast. We see that the efficiency is indeed higher under both versions of Bartercast, but only minimally so (less than 5% difference).

The more interesting analysis concerns the overall efficiency with strategic agents present. The *efficiency* of a particular agent type is defined to be the average amount of

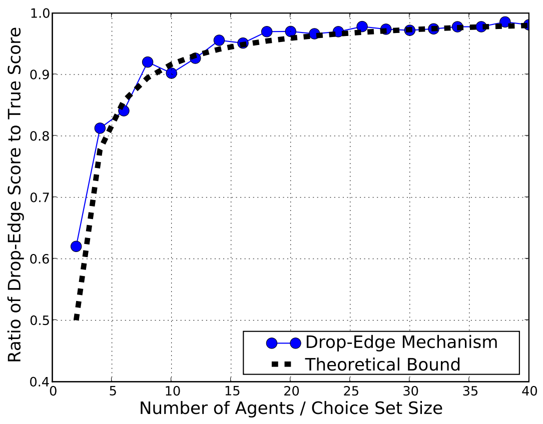
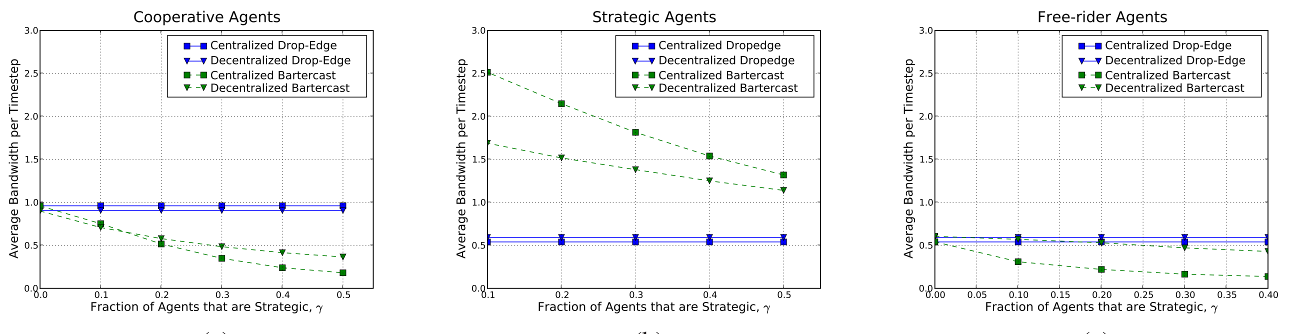


Figure 3: The approximation ratio of the scores.



(a) (b) (c)

Figure 4: Bandwidth of different types of agents, using Bartercast and Drop-Edge. The fraction of free-riding agents is $\beta = 0.5$.

bandwidth received by that type of agent per time step. *It is our goal to maximize the efficiency of the cooperative agents and to minimize the efficiency for free-riding agents, and for strategic free-riders in particular.* Ultimately, the goal is to cause agents to change from free-riding to cooperating.

We compare Figures 4 (a),(b) and (c), to analyze the relative efficiency of all agent types under the Bartercast and Drop-Edges mechanisms. Note that the total efficiency is the same for both mechanisms because the amount of work performed by individual agent types is fixed. In Figure 4(b), we clearly see that strategic agents are able to sharply increase their performance compared to the other agents (see Figures 4(a) and (c)) by misreporting under the BarterCast mechanism. 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. With BarterCast, agents have a very large incentive to act strategically. The Drop-Edge mechanism in contrast leads to the same constant efficiency for each individual agent type (because there is no incentive to manipulate), and in particular the efficiency of cooperative agents is almost twice as high as that of free-riding agents.

In practice, strategic misreports may also occur under Drop-Edge even though such behavior is not rational for an individual agent. We have tested Drop-Edge in settings with strategic agents (not plotted) and although the efficiency of the cooperative agents decreases slightly as the proportion of strategic agents increases, Drop-Edge continues to clearly outperform Bartercast. Interestingly, if we move from a 1-hop maxflow to a standard maxflow algorithm, the efficiency of cooperative agents under Drop-Edge actually increases as the proportion of strategic agents increases. A more detailed analysis of this effect is subject to future investigations.

We also ran a longer experiment with $\beta = 0.5, \gamma = 0.2$ for 500 time steps, measuring how efficiency changes over time. In Figure 5, we see that the benefit that strategic agents gain from misreporting in BarterCast gets even larger over time. Compare this against Figure 6, which presents results for DropEdge. Strategic agents cannot manipulate their scores, and receive decreasing amounts of work as the simulation proceeds. At the end of the run, cooperative agents indeed receive twice as much work per round as the other agents (note they also perform exactly twice as much work).

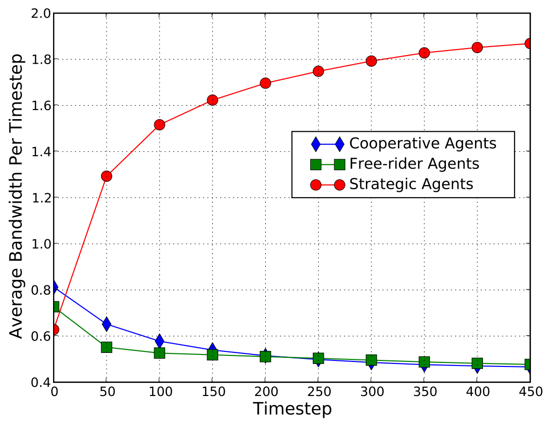


Figure 5: Bandwidth in BarterCast Mechanism over Time.

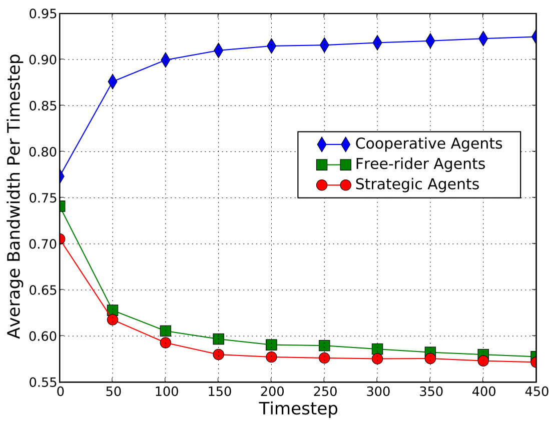


Figure 6: Bandwidth in Drop-Edge Mechanism over Time.

6 Ongoing & Future Research

We are currently pursuing multiple avenues to further improve the design of work accounting mechanisms. First, we want to test the performance of the Drop-Edge Mechanism in a more realistic environment. To this end we have partnered with the authors of BarterCast (Meulpolder et al. 2009), and we are currently implementing the Drop-Edge Mechanism into the Tribler BitTorrent client. This allows us to run trace-based simulations on the piece level. During this transition from a very stylized simulation to a real-world implementation, we realized quickly that now the computational complexity of the Drop-Edge mechanism plays a much larger role. In fact, when comparing BarterCast to Drop-Edge, one drawback of Drop-Edge becomes apparent: every time the composition of the choice set changes, the agents have to re-compute the max-flow algorithm which is a costly operation. Obviously, this was not necessary for the BarterCast mechanism. But in fact, if both mechanisms are implemented as described in Section 3, then any time an edge in the subjective work graph changes its weight, the scores would need to be re-computed. To avoid running the full max-flow algorithm each time a change occurs, we are currently investigating incremental max-flow algorithms that have better running times.

The second avenue involves a more detailed analysis of the max-flow algorithm that is used in both BarterCast and Drop-Edge. Earlier in the paper, we have argued that using max-flow to estimate an agent's net work is a form of "bounded addition". A closer look at the semantics of the max-flow algorithm reveals, that simply taking the flow between two agents in one direction minus the flow in the other direction does not really make sense. By using max-flow we are limiting the influence of any agent on the max-flow path by the incoming and outgoing edges of that agent respectively. However, this only makes sense for an agent's outgoing edges which represent work it has performed for other agents, and thereby earned trust and is now allowed to make reports about other agents. However, incoming edges represent work consumed and thus there is no reason we should trust an agent more when the agent has a high weight on incoming edges. It is easy to show that a Byzantine agent that never performs any work but manages to consume work is now able to spread bad information in the accounting mechanism despite the use of the max-flow algorithm. We are currently exploring variations of the max-flow algorithm that do not suffer from this problem.

The third avenue regards sybil attacks on work accounting mechanisms. Despite using the max-flow algorithm which provides sybilproofness in reputation systems (Cheng and Friedman 2006), both BarterCast and Edge-Drop can be manipulated by sybil attacks. This is because sybil attacks are more powerful against work accounting mechanisms than against reputation systems. In PageRank for example, the only concern about a sybil attack is that an agent could increase the reputation of its website by creating a set of sybils that are linking to the original website but an agent does not directly benefit from a sybil website with a high reputation. Various reputation mechanisms (e.g., maxflow, hitting-time (Cheng and Friedman 2006; Sheldon and Hopcroft 2007) are sybil-proof in the sense that they can protect against these kinds of attacks. The situation is drastically different in distributed work systems. For example, in P2P file-sharing, if I can create sybils with a positive score, then I can use these sybils to receive work from other users without any negative effect on my real score. Existing sybil-proof mechanisms do not protect against these kinds of sybil attacks. Thus a new approach is required and subject to our ongoing research.

7 Conclusion

In this paper we have formalized the problem of designing incentive-compatible accounting mechanisms for distributed work systems. We have shown that the existing BarterCast mechanism is highly susceptible to misreport attacks and that strategic agents could benefit significantly from manipulating. To address this, we have introduced the Drop-Edge Mechanism which is misreport-proof. In our theoretical analysis we have proved that Drop-Edge provides good approximations to an agent's net contributions and scores respectively, and is accurate as the number of agents grows. Via simulations of a P2P file-sharing system we have shown that the good approximation of the scores also translates into good system efficiency. When strategic agents are present, the Drop-Edge Mechanism clearly outperforms BarterCast: cooperative agents have higher efficiency, while free-riding agents have lower efficiency. We have shown that the magnitude of this effect even grows over time. Thus, we believe that using Drop-Edge over BarterCast in a real system has significant advantages.

Acknowledgements

We are thankful to Michel Meulpolder, Johan Pouwelse, Ian Kash and Ariel Procaccia for helpful discussions on this work. The first author is supported by a Microsoft Research PhD Fellowship. The second author is supported by an NDSEG Fellowship sponsored by the Department of Defense.

References

Alon, N.; Fischer, F.; Procaccia, A. D.; and Tennenholtz, M. 2010. Sum of us: Strategyproof selection from the selectors. *Working Paper, Harvard University*.

Cheng, A., and Friedman, E. 2006. Manipulability of PageRank under Sybil Strategies. In *Proceedings of the First Workshop of Networked Systems (NetEcon06)*.

Feldman, M.; Lai, K.; Stoica, I.; and Chuang, J. 2004. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*.

Friedman, E. J.; Halpern, J. Y.; and Kash, I. 2006. Efficiency and Nash Equilibria in a Scrip System for P2P Networks. In *Proceedings of the 7th ACM Conference on Electronic Commerce (EC)*.

Friedman, E.; Resnick, P.; and Sami, R. 2007. *Algorithmic Game Theory*. Cambridge University Press. chapter Manipulation-Resistant Reputation Systems, pp. 677–698.

Meulpolder, M.; Pouwelse, J.; Epema, D.; and Sips, H. 2009. 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)*.

Piatek, M.; Isdal, T.; Krishnamurthy, A.; and Anderson, T. 2008. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 1–14.

Pouwelse, J.; Garbacki, P.; Epema, D.; and Sips, H. 2005. The Bittorrent P2P File-Sharing System: Measurements and Analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*.

Resnick, P., and Sami, R. 2009. Sybilproof transitive trust protocols. In *Proceedings of the Conference on Electronic Commerce*.

Sheldon, D., and Hopcroft, J. 2007. Manipulation-Resistant Reputations Using Hitting Time. In *5th Workshop on Algorithms for the Web-Graph*.