Work Accounting Mechanisms: Theory and Practice*

Sven Seuken†, Michel Meulpolder‡, Dick H. J. Epema‡,
David C. Parkes†, Johan A. Pouwelse‡, Jie Tang§

December 15, 2010

PRELIMINARY DRAFT!

Abstract

The Internet has enabled a new paradigm of economic production, where individual users perform work for others, often in small units, for short periods of time, and without formal contracts or monetary payments. These *distributed work systems* can arise in many places, for example in peer-to-peer (P2P) file-sharing networks, in ad-hoc wireless routing networks, or in 3G content distribution systems. The particular challenge is to incentivize users to perform work for others, even though all interactions are bilateral and monitoring is not possible. In this paper, we formalize the problem of designing *incentive-compatible work accounting mechanisms* that measure the net contributions of users, despite relying on voluntary reports. We describe a fully distributed accounting mechanism called BarterCast and introduce the Drop-Edge extension which removes any incentive for a user to make misreports about its own interactions. We prove that the information loss necessary to achieve this incentive compatibility is small and vanishes in the limit as the number of users grows. In some domains, users may be able to cheaply create fake identities (i.e, sybils) and use those to manipulate the accounting mechanism. A striking negative result is that no sybil-proof accounting mechanism exists if one requires responsiveness to a single positive report. To evaluate the welfare properties of BarterCast+DropEdge, we first present results from a discrete, round-based simulation, showing that the mechanism achieves very high efficiency. We have also implemented the mechanism in Tribler, a BitTorrent software client, that is already deployed in the real world and has thousands of users. Experimental results using Tribler demonstrate that the mechanism successfully prevents free-riding in P2P-file-sharing systems, and achieves better efficiency than the standard BitTorrent protocol.

Keywords: Mechanism Design, Distributed Systems, Accounting.

*We thank seminar participants from the Harvard EconCS group and the AAAI'10 conference for very useful feedback. Seuken gratefully acknowledges the support of a Microsoft Research PhD Fellowship. This paper is an extended version of two conference papers [22], [16].

†School of Engineering & Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA 02138, {seuken, parkes}@eecs.harvard.edu.

‡Department of Computer Science, Delft University of Technology, the Netherlands, {m.meulpolder, D.H.J.Epema, j.a.pouwelse}@ewi.tudelft.nl

§Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94720, jietang@eecs.berkeley.edu

1 Introduction

Distributed work systems arise in many places, for example in peer-to-peer (P2P) file-sharing networks, where users share videos, music or software with each other, or in ad-hoc wireless networks where individual peers route data packages for each other through the network. Amazon Mechanical Turk and other "crowd sourcing" applications suggest an explosion of interest in this distributed form of 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 amount of work performed by a population must equal the total amount of 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. Otherwise, a majority of the agents may seek to free-ride on the system, i.e., minimize the work they perform for others and maximize the work they can consume. Current systems often enforce temporally-local balances, e.g., via fair exchange protocols where agent A only continues to perform work for agent B if agent B reciprocates immediately. 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 [17]. For example, [18] found that more than 80% of BitTorrent users go offline immediately once they have finished downloading. *Accounting mechanisms* would solve this problem by keeping long-term tallies of work performed and consumed by each user. This would give users an incentive to share even after they have finished downloading, and thus increase system efficiency.

The particular challenge that we address here occurs when the interactions are bilateral and there is no ability for a third party to monitor the activities. We consider distributed work systems where the agents perform small units of work for each other, for limited periods of time, such that no contract covers the interaction, because the effort for creating and legally enforcing the contracts would be too high relative to the value of the work. This rules out the use of any kind of real or virtual currency, because the institutional requirements for the exchange of payments are not available. Furthermore, we assume that there is no a-priori trust relationship between the agents in the network that could be used, but instead, that an agent can only earn trust by performing work. Because there is no center monitoring the interactions, the accounting mechanism must rely on voluntary reports by the same users that perform and consume work in the system. Thus, an accounting mechanism must be robust to manipulations, because users might otherwise overstate the amount of work contributed, or understate the amount of work consumed. It is our goal to design accounting mechanisms that are *incentive-compatible* in the sense that it is in each individual user's best interest to make truthful reports about his interactions with other users.

1.1 Accounting vs. Reputation Mechanisms

The problem of designing an incentive-compatible accounting mechanism shares some features with work on trust/reputation mechanisms [9]. 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.

1.2 Real-world applications for Accounting Mechanisms

The most-widely used application that falls into the category of distributed work systems we study is peer-to-peer file-sharing. The performance of P2P file sharing systems crucially depends on the contribution of resources by their participating users. None of the existing protocols provides long-term sharing incentives to its users. Free-riding is a well-known issue in P2P research; its effects have been empirically measured [1, 21] and extensively analyzed [3, 7, 13, 15]. None of the existing systems provides the users with a long-term incentive to upload data to other users. Thus, P2P file-sharing could benefit a lot from a well-designed accounting mechanism.

Another interesting application for accounting mechanisms is content distribution in 3G networks. 3G bandwidth is a very scarce resource, downloading data via a 3G network is relatively slow and requires a lot of battery power from a smartphone or similar device. In contrast, the supply of bandwidth is wireless networks is relatively cheap, the download speed in such networks is very fast, and smartphones requires much less battery power to connect to a wireless network than to a 3G network. Recently, researchers have started to look into the potential of using ad-hoc wireless networks to distribute certain data instead of using 3G bandwidth [4]. Imagine, you are at the train station at 9am in the morning, and lots of people are downloading the news, the weather, etc. onto their smartphones. Whenever multiple near-by users are downloading the same data, the efficiency could be increased, if only one of the users downloads the data via the 3G network, and then the data is distributed via ad-hoc wireless networks to the other users. This is already technically feasible and can happen without any user interaction. However, given that 3G connections are costly for the user (draining the battery, and accumulating MBs that count towards the monthly download limit), with a naive implementation, no users would like to be the one who downloads the data packages via 3G and then distributes them. By using an accounting mechanism like the one we propose in this paper, we can solve this problem, balancing the work load over time, and giving each user an incentive to consume and perform work at different points in time.

A third application concerns routing in ad-hoc networks. Imagine you need to set up a communication network in an area without existing communication infrastructure, for example, in an area that has just been hit by a natural disaster or in a war zone. In such situations, oftentimes different organizational units, potentially from different countries, set-up camps next to each other, but have no direct relationship with each other. Ad-hoc wireless networks are a very efficient way of quickly establishing a communication network in these situations, however, bandwidth will generally be scarce, and routing data packages from other organization units through your own network might reduce the amount of bandwidth you can use yourself. Again, a properly designed accounting mechanism can address the problem by making sure that over time, each networks routes approximately the same number of data packages, thereby providing the different network operators with an incentive to collaborate in the set-up of these networks.

1.3 Outline and Overview of Results

In this paper we present a theoretical and experimental analysis of accounting mechanisms for distributed work systems. In Section 2 we formalize the concept of a distributed work system and introduce BarterCast, a fully decentralized, lightweight information exchange system. In Section 3 we present what we believe to be the first formal model for the design of incentive compatible accounting mechanisms. We show how the BarterCast-basic mechanism is susceptible to misreport manipulations, and we introduce the Drop-Edge extension that removes any incentives for agents to misreport information, by selectively dropping information dependent on the decision context. We provide a theoretical analysis of Drop-Edge in Section 4.1 where we show that the information loss due to dropping some of the information is small and vanishes in the limit as the number of agents in the network gets large. In Section 4.2 we analyze the problem of sybil attacks on accounting mechanisms. We first show that, if we require that a single positive report about an agent shall make that agent better off than a new-comer, then no sybil-proof accounting mechanism exists. However, we show how we can design accounting mechanisms that are sybil-proof against certain kinds of attacks when we relax this requirement. In Section 5 we provide an extensive experimental evaluation of the BarterCast-DropEdge mechanism in a discrete simulation environment. First, we show that the approximation ratio of Drop-Edge is very good in practice. Next, we show efficiency results comparing BarterCast-basic and BarterCast-Drop-Edge. We show convincingly that strategic agents that manipulate the accounting mechanism benefit significantly in BarterCast-basic such that the performance of cooperative agents suffers significantly. In contrast, BarterCast-Drop-Edge is robust against misreport attacks, and thus strategic agents are not doing any better than standard free-riders, and cooperative agents have a significantly higher efficiency. We also show that this effect increases over time, as agents gather more and more information about each other and thus are able to discriminate better and better between sharers and free-riders. In Section 6 we provide results from extensive experiments in the BitTorrent P2P file-sharing network. We have already implemented a series of accounting mechanisms into Tribler, a real file-sharing client that is already deployed and being used by thousands of users. Using Tribler we ran simulation experiments on the BitTorrent protocol level, giving us much more realistic simulation results about the performance of various accounting mechanisms. In Section 6.1 we explain why the requirement of being backwards-compatible introduces new difficulties for making use of accounting mechanisms in BitTorrent, and in Section 6.3.2 we discuss in detail the advantages and disadvantages of using a ranking policy or a banning policy for making work allocation decisions in BitTorrent. We show that using the ranking policy, the power of accounting mechanisms is inherently limited in BitTorrent. However, we show that using the banning policy, we can achieve performances for sharers and free-riders that different significantly (by more than a factor of 2), and thus a behavioral change can be expected (where free-riders become sharers after a while). In Section 6.3.5 we discuss in detail the effects of accounting accuracy and noise in large networks and explain why we can expect high accuracy and low noise in real BitTorrent networks. Finally, in Section 6.4, we present experimental results that are trace-based, thus, using an agent-arrival process that we have actually observed on a real-world BitTorrent tracker. These experiments verify our previous results, showing that the agents' accounting scores have a very high positive correlation with their true net contributions to the network. Furthermore, all experiments show that using the BarterCast-Drop-Edge mechanism, we can separate the performance of sharers and free-riders significantly, and assuming some kind of behavioral change (i.e., free-riders becoming sharers over time), we can achieve a total efficiency that is higher than with the standard BitTorrent protocol.

1.4 Related Work

Most of the prior work related to distributed work systems and accounting mechanisms has studied the domain of P2P file-sharing, often with the primary goal to prevent free-riding. Gupta et al. [11] present a reputation system that is partially distributed, but relies on the authority of a single agent that stores peer reputations. As a mechanism for multiple agents is not discussed, the presented solution in essence remains centralized. Kamvar et al. [12] present EigenTrust, an algorithm for reputation management in P2P networks that is based on distributed computing of a globally consistent trust vector for every peer in the system. EigenTrust's theoretical properties are attractive. On the downside, EigenTrust relies on pre-trusted peers for convergence and its aim for global consistency assumes a rigid network of peers. Karma [24] is a system in which sets of bank nodes keep track of the transaction balance of peers. The approach relies on a DHT-based structure of allocating bank nodes and cryptographic mechanisms for security. BitTorrent [6] uses a tit-for-tat policy with short-term, direct incentive properties. While tit-for-tat is successful in short term transactions, the mechanism offers no incentives for long-term sharing of content. Lian et al. [14] identify key disadvantages of both EigenTrust and tit-for-tat and propose a hybrid model in between the two.

Piatek et al. [17] 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. [7, 8] study the challenges involved in providing robust incentives in fully decentralized P2P networks, including free-riding, whitewashing, and misreport attacks. They introduce a reputation mechanism based on maxflow, but they implicitly assume that all nodes have a complete view of the network, which is highly unrealistic in large, dynamic communities. Moreover, the authors omit any discussion regarding how nodes acquire this information. Furthermore, in contrast to our proposal, their mechanism is not misreport-proof. 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 only consider one-shot games, and don't consider the problem of agents earning trust over time. A recent paper by Resnick and Sami (2009) addresses specifically the question of transitive trust, 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 [10]. There, work provision or consumption is observable and there is a trusted currency. In general, we are not aware of any practically feasible mechanism that is decentralized to the same degree as our proposal, misreport-proof, tested under realistic, trace-based community conditions, and deployed in a real-world P2P network.

2 Distributed Work Systems

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.*

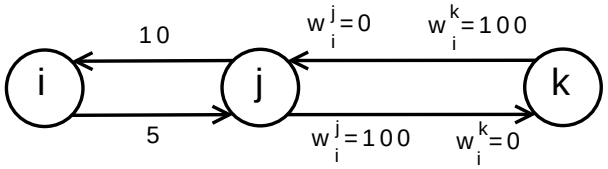


Figure 1: A subjective work graph from agent i's perspective. Edges where i has direct information have only one weight. Other edges can have two weights, corresponding to the possibly conflicting reports of the two agents involved.

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.*

We assume that agents can share their information with each other, either via a centralized or a decentralized message exchange system. Based on its own experiences and the reports it has received 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 weights 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 $w(j,k)$ and thus can possibly be in conflict with each other, even if they have been submitted at the same point in time. We now describe in more detail how agents can exchange information with each other, to obtain the information necessary to construct the subjective work graph.

Throughout the paper, we analyze and compare two different modes of information sharing between the agents: *centralized* and *decentralized* information exchange. For centralized information exchange, we assume the existence of a center (e.g., a centralized server on the Internet). After every interaction, each agent makes a report to the center which stores all reports persistently. At any point in time, an agent can query the center to obtain the most up-to-date information about all reports available at the center to then construct its subjective work graph based on his own information and the information obtained from the center. Note that every agent has different *private* information about his own interactions, and the other information is not necessarily correct, because agents can make false reports to the center. Thus, the resulting subjective work graphs for the agents can differ, even if all the reports are stored at the center. However, if agents i and j both report truthfully to the center, then their respective subjective work graphs G_i and G_j are the same.

In many environments, a centralized information exchange system is simply not feasible, for example in wireless ad-hoc networks. In other environments, a centralized system might not be desirable for many reasons: a center represents a single point of failure, a center presents a bandwidth bottleneck, and a center requires some a priori trust in one entity and it is unclear how that trust should be established. This motivates the study of fully *decentralized information exchange systems.*

We now present BarterCast, a fully decentralized, lightweight information exchange system. In BarterCast, each agent also keeps a private history of its direct interactions with other agents, but

Figure 2: Accounting Mechanism and Allocation Policy: based on the subjective work graph G_i and the current choice set C_i, the accounting mechanism computes a score $S_j^M(G_i, C_i)$ for each agent in the choice set. Based on these scores, the allocation policy selects one agent for whom agent i will perform work.

in addition, an agent obtains information about the rest of the network by exchanging a selection of its private history with others using bilateral messages. We assume that peers can discover other peers with whom to exchange messages by using a *Peer Sampling Service*. When two agents agree to exchange messages, then agent i selects for its messages the records of the N_h agents with the highest amount of work performed for i as well as the N_r agents most recently seen by i. Thus, in contrast to a centralized information exchange system, in BarterCast, each individual agent will always only have an incomplete view of the whole network, and a significant part of the information it holds will not be up-to-date. However, we will show in Section 5 that despite these drawbacks, the subjective work graphs each agent constructs based on the information available via BarterCast are sufficient to distinguish sharers from free-riders.

3 Accounting Mechanisms

3.1 Preliminaries

In the distributed work system, at every point in time, an agent can decide whether he is willing to perform work for others or not. An agent who makes himself available to perform work receives work requests by a set of agents (with which the agent may have rarely or never interacted with before). For example, in a P2P file-sharing application, each agent that has any pieces of a particular file will be contacted by a group of agents that are all interested in some of those pieces. At any moment in time, the contacted agent will have to choose for whom to perform work from this set of agents.

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.*

With agents $j, k \in C_i$, agent i must now decide whether he should do work for agent j or rather do work for another agent k. We assume that an agent has no *a priori* bias towards assisting one agent over another. We would like to know whether agent j is a "good" agent, i.e., whether he has done approximately as much work as he has received, or whether he is a "bad" agent that tries to free-ride on the system. The role of an accounting mechanism is to compute a *score* for each agent[1], proportional to the net work contributed by each agent $j \in C_i$ to the system to differentiate between "good" and "bad" agents.

[1]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.

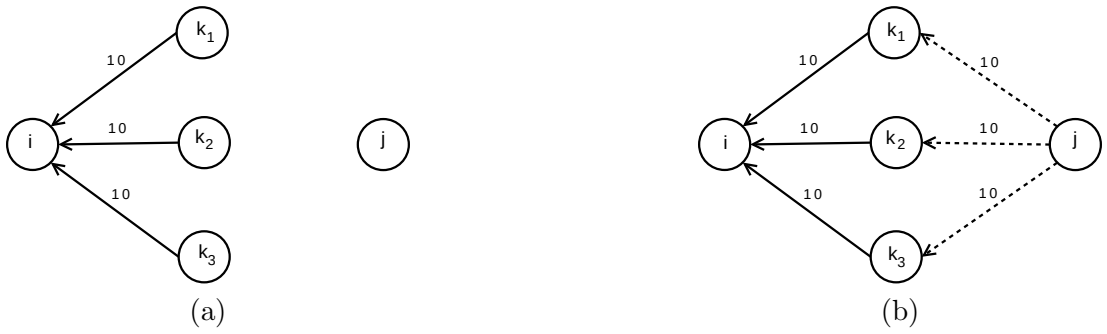


Figure 3: (a) A work graph based on true reports. (b) The subjective work graph as viewed by i, including a misreport attack by j to boost its score in BarterCast. Dotted edges indicate misreports.

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.*

Once the accounting mechanism has computed a score for each agent in the choice set, the agent uses an *allocation policy* to decide who to allocate work to (see Figure 2). Thus, the accounting mechanism together with the allocation policy can be seen as a matching algorithm, allocating work-performing agents to work-seeking agents. In this paper, we consider the following two allocation policies:

Definition 6. ***(Ranking Policy)*** *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.*

Definition 7. ***(Banning Policy)*** *Given subjective work graph G_i, choice set C_i, accounting mechanism M, and a banning threshold δ, agent i performs one unit of work for an agent chosen uniformly at random from $\{j \in C_i | S_j^M(G_i, C_i) \geq \delta\}$.*

Other allocation policies are also conceivable, e.g., a proportional allocation policy, but in this paper, we only compare the ranking and the banning policies. We will perform a more detailed comparison of the two policies in Section 6, where we show the advantages and disadvantages of either policy in the P2P file-sharing domain.

3.2 Agent Population and 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. We also consider a second kind of attack on the accounting mechanism called *sybil attacks* where an agent inserts fake agents into the network to manipulate the mechanism. However, we defer a discussion of these attacks to Section 4.2.

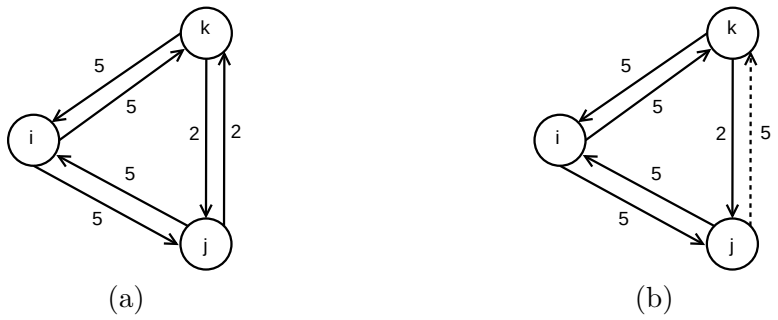


Figure 4: (a) A Work graph based on true reports. (b) The subjective work graph as viewed by i, including a misreport attack by j to decrease k's score and increase its own score.

Definition 8. ***(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*
- $S_k^M(G_i', C_i) \geq S_k^M(G_i, C_i)\ \forall k \in C_i \setminus \{j\}$.[2]

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.

3.3 BarterCast Basic

Definition 9. ***(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 $S_j^B(G_i, C_i) = MF_{G_i^B}(j,i) - MF_{G_i^B}(i,j)$.[3]

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

[2]Note that the first requirement is equivalent to *value-strategyproofness* as defined for trust mechanisms, and both requirements together imply *rank-strategyproofness* [5].

[3]This specification of BarterCast differs from Meulpolder et al. [16] 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.

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 Figures 3 and 4. We always show the subjective work graph from i's perspective and the manipulating agent is j. Figure 3 (a) shows a true work graph. Figure 3 (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 Figure 4 (a) which shows a new true work graph. Figure 3 (b) 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.

3.4 BarterCast + DropEdge

Definition 10. ***(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 $S_j^D(G_i, C_i) = MF_{G_i^D}(j,i) - MF_{G_i^D}(i,j)$. [4]

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

4.1 Information Loss of Drop-Edge

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

[4] 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.

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.

$$\begin{aligned}
&\mathbb{E}_C[work_i(k, G_i^D)] && (4)\\
&= \mathbb{E}_C[\sum_{j \neq k} \tilde{w}_i^D(k, j)] && (5)\\
&= \sum_{j \neq k} \mathbb{E}_C[\tilde{w}_i^D(k, j)] && (6)\\
&= \sum_{j \neq k} \Big[\frac{(m-1)}{(n-1)} \cdot 0 + \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot \tilde{w}_i^O(k, j) \Big] && (7)\\
&= \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot work_i(k, G_i^O)
\end{aligned}$$

For equation (7), 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. □

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.[5] 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, $S_{j,h}^O(G_i, C)$ is the score computed by an omniscient mechanism for exactly h hops.

[5] 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.

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 $k \in C$:*

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

Proof.

$$E_C[S_k^D(G_i, C)] = E_C[MF_{G_i^D}(k, i) - MF_{G_i^D}(i, k)] \tag{8}$$

$$= \sum_{h=0}^{n-m-1} E_C[MF_{G_i^D,h}(k, i) - MF_{G_i^D,h}(i, k)] \tag{9}$$

$$= E_C[MF_{G_i^D,0}(k, i) - MF_{G_i^D,0}(i, k)] \tag{10}$$

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

$$= S_{k,0}^O(G_i, C) \tag{12}$$

$$+ E_C[MF_{G_i^D,1}(k, i) - MF_{G_i^D,1}(i, k)] \tag{13}$$

$$+ \sum_{h=2}^{n-m-1} E_C[MF_{G_i^D,h}(k, i) - MF_{G_i^D,h}(i, k)] \tag{14}$$

$$= S_{k,0}^O(G_i, C) + \left(\frac{n-m-1}{n-2}\right) \cdot S_{k,1}^O(G_i, C) \tag{15}$$

$$+ \left(\frac{n-m-1}{n-2}\right) \cdot \left(\frac{n-m-2}{n-3}\right) \cdot S_{k,2}^O(G_i, C) \tag{16}$$

$$+ \sum_{h=3}^{n-m-1} E_C[MF_{G_i^D,h}(k, i) - MF_{G_i^D,h}(i, k)] \tag{17}$$

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

In Equation 10 we isolated the expectation of the 0-hop max-flow terms, which are direct paths between i and k and thus the edges are not dropped and Equation 12 follows. In Equation 13 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 15 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 $k \in C$:*

$$E_C[S_{k,H}^D(G_i, C)] \geq \left(\frac{n-m-H}{n-H}\right)^H S_{k,H}^O(G_i, C).$$

In the BarterCast implementation [16] 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:

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

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 Sections 5 and 6 show that the information-loss of Drop-Edge is small and that the mechanism indeed performs very well in practice.

4.2 Sybil-proofness

Now we turn our attention to a class of attacks called *sybil manipulations*, where an agent introduces sybil nodes (fake agents) into the network to manipulate the accounting mechanism. For now, we will only consider sybil agents that do not perform work themselves, but only try to consume work from other agents.

Definition 11. *(Sybil Manipulation) Given a work graph $G_i = (V_i, E_i, w_i)$, a sybil manipulation by agent $j \in V_i$ is a tuple $\sigma_j = (S, E_s, w_s)$ where $S = \{s_{j_1}, s_{j_2}, ...\}$ is a set of sybil agents created by agent j, $E_s = \{(x, y) : x, y \in S \cup j\}$ is a set of edges and w_s are the edge weights for the edges in E_s. Applying the sybil manipulation σ_j to G_i results in a modified work graph $G_i \downarrow \sigma_j = G'_i = (V_i \cup S, E_i \cup E_s, w')$ where $w'(e) = w_i(e)$ for $e \in E_i$ and $w'(e) = w_s(e)$ for $e \in E_s$.*

We let S^M_0 denote the default score that accounting mechanism M assigns to an agent about which no information regarding work consumed and work performed is available. Note that the analysis of a sybil manipulation does not require a dynamic, multi-step analysis. Even though an attacking agent could do multiple things, e.g., add sybils to the network, make multiple false reports about these sybils, all of these are modeled as happening in one step. The analysis only consider a graph G before the sybil manipulation, and a graph G' after the manipulation.

Definition 12. *(Sybil-Proofness) An accounting mechanism M is sybil-proof if for any subjective work graph $G_i = (V_i, E_i, w_i)$ and choice set C_i, for any agent j with any sybil manipulation $\sigma_j = (V_s, E_s, w_s)$ such that $G_i \downarrow \sigma_j = G'_i$, the following three properties hold:*

(1) $S^M_j(G_i, C_i) \geq S^M_j(G'_i, C_i)$

(2) $\forall k \in V_i \setminus \{j\} : S^M_k(G_i, C_i) \leq S^M_k(G'_i, C_i)$

(3) $\forall s \in S : S^M_s(G'_i, C_i) \leq S^M_0$

i.e., (1) j cannot increase its own score, (2) j cannot decrease another agent's score, and (3) j cannot create a sybil node that has a higher score than the default score. It is easy to see that BarterCast and BarterCast+DropEdge are not sybil-proof mechanisms. Consider Figure 5 where agent j performs a successful sybil manipulation.

We make the assumption that the scores an accounting mechanism computes only depend on the amount of work performed and consumed by the agents in the network. More specifically, we assume that adding or removing agents with no amount of work consumed or performed does not changes the scores of other agents:

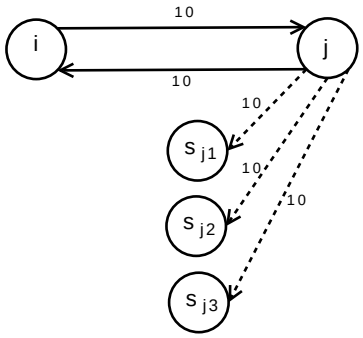


Figure 5: A sybil attack where agent j generates many sybils, then does a little bit of work for i and then provides its sybils with positive reputation.

Assumption 1. *(Independence of Disconnected Agents)) Given any subjective work graph $G_i = (V_i, E_i, w_i)$ and any choice set C_i, for any $k \in V_i$ for which there does not exist an edge in E_i or for which all edges in E_i have zero weight, let $G_i' = (V_i', E_i', w_i')$ denote the graph where node k has been removed, i.e., $V_i' = V_i \setminus \{k\}$, $E_i' = E_i \setminus \{(x, y) : x = k \vee y = k\}$, and $w_i'(e) = w_i(e)$ for all $e \in E_i'$, then we assume that the following holds:*

$$\forall j \in V_i' : S_j^M(G_i, C_i) = S_j^M(G_i', C_i')$$

We assume that, a priori, the accounting mechanism does not put more or less trust into any agent in the network. We will only consider mechanisms which, for any renaming of the agents in the network, return the same scores:

Definition 13. *(Symmetric Accounting Mechanisms) An accounting mechanism M is symmetric if, for any node i with any any subjective work graph $G_i = (V_i, E_i, w_i)$ and choice set C_i, any graph isomorphism f such that $G_i^{'} = f(G_i)$, $C_i^{'} = f(C_i)$ and $f(i) = i$:*

$$\forall j \in V_i \setminus \{i\} : S_j^M(G_i, C_i) = S_{f(j)}^M(G_i^{'}, C_i^{'}).$$

We will exclude from our analysis any "trivial" accounting mechanisms, which excludes those mechanisms that assign the same score to every agent or that assign random scores to every agent. But we will also exclude mechanisms that ignore all information except for their own direct experiences. We begin our formal analysis of sybil-proof accounting mechanisms, assuming that even one report from an agent can change the score of another agent. To make this more formal:

Definition 14. *(Single-Report Responsiveness Property) Let $dist(i, j)$ denote the distance between two nodes i and j in a graph, i.e, the length of the shortest path between those two nodes. An accounting mechanism M has the single-report responsiveness property if, for any agent i, there exists a subjective work graph $G_i = (V_i, E_i, w_i)$ and choice set C_i, with nodes j and k such that $dist(i, j) = dist(j, i) = 1$ and $dist(i, k) = dist(k, i) = \infty$ (i.e., nodes i and j are neighbors in G_i and there is no path connecting nodes i and k), and there exists a graph $G_i' = (V_i', E_i', w_i')$ with $V_i' = V_i$, $E_i' = E_i \cup \{(k, j), (j, k)\}$, and $w_i'(e) = w_i(e)$ for all $e \in E_i \setminus \{(k, j), (j, k)\}$, and there exists a constant $c \in \mathbb{R}_{>0}$ with $w_i'^{j}(k, j) = c$, such that:*

$$S_k^M(G_i', C_i') > S_0^M$$

What the single-report responsiveness property says is that there exists a situation, where a single positive report by agent j about agent k increases the score that agent i assigns to agent k above the default score.

Note that BarterCast-basic and BasterCast+DropEdge are both symmetric mechanisms that satisfy the single-report responsiveness property. We have already shown that both mechanisms are susceptible to sybil attacks. We will now show that this is generally unavoidable:

Theorem 3. *No accounting mechanism M that satisfies Assumption 1, is symmetric and has the single-report responsiveness property is sybil-proof.*

Proof. Let's assume accounting mechanism M satisfies the independence assumption (Assumption 1), is symmetric and has the single-report responsiveness property. Thus, there exists a graph G_i and nodes i, j and k as described in Definition 14. Now, let agent j create a sybil node s_j and insert it into G_i such that $G'_i = (V'_i, E_i, w_i)$ with $V'_i = V_i \cup \{s_j\}$. Because of the independence assumption, the scores of all agents in the graph have remained the same. Note that there is no path connecting k and i as well as no path connecting s_j and i, and thus the two nodes k and s_j look the same from i's perspective. Because M is symmetric, we could apply a graph isomorphism f to G'_i that only switches the labeling of s_j and k but nothing else. Thus, from the single-report responsiveness property, we can now conclude that there exists a report that j can make about s_j with $w'^{j}_{i}(s_j, j) = c$ leading to graph G''_i such that $S^M_{s_j}(G''_i, C_i) > S^M_0$. This violates property (3) of sybil-proofness in Definition (12), and thus M is not sybil-proof. □

One might argue that, even though an accounting mechanism M is not sybil-proof, all possible sybil manipulations that it allows could be *non-beneficial* for the manipulating agent. For example, it is conceivable that, even though an agent j might be able to create sybils with a positive score under mechanism M, the mechanism would in return decrease agent j's score in such a way, that overall, the sybil manipulation would not be beneficial for j. We will show next that, if we also assume that the accounting mechanism is misreport-proof, then every mechanism with the above properties has a beneficial sybil manipulation. Let's first formalize what we mean by a *beneficial sybil manipulation*:

Definition 15. *(Beneficial Sybil Manipulation) Given any work graph $G_i = (V_i, E_i, w_i)$, a beneficial sybil manipulation σ_j by agent $j \in V_i$ such that $G'_i = \sigma_j(G_i)$ is one where property (1), (2) or (3) holds:*

(1) $\exists C_i$ *s.t.* $j \in C_i$ *and* $S^M_j(G_i, C_i) < S^M_j(G'_i, C_i)$

(2) $\exists k \in V_i \backslash \{j\}$ *and* $\exists C_i$ *with* $j, k \in C_i$*:* $(S^M_j(G_i, C_i) < S^M_k(G_i, C_i)) \quad \wedge \quad (S^M_j(G'_i, C_i) > S^M_k(G'_i, C_i))$

(3) $\exists s \in S : (\exists C_i$ *with* $s \in C_i : S^M_s(G'_i, C_i) > S^M_0) \quad \wedge \quad (\forall C_i$ *with* $j \in C_i : S^M_j(G'_i, C_i) \geq S^M_j(G_i, C_i))$

i.e., agent j can (1) increase its own score, or (2) affect its own score and that of another agent in such a way that the relative ranking of the two agents changes, or (3) create a sybil agent with a positive score while non-decreasing its own score. Note that manipulations (1) and (2) are the same ones that we worry about when designing reputation mechanisms. However, manipulation (3) is unique to the design of accounting mechanisms.

Theorem 4. *For every accounting mechanism M that satisfies Assumption 1, is symmetric, has the single-report responsiveness property and is misreport-proof, there exists a beneficial sybil manipulation.*

Proof. Consider the same set-up as in the proof for Theorem 3, with agents i, j, and k. Now, assume that agent k performs c units of work for j and agent j makes a truthful report to i about this interaction, leading to subjective work graph G'_i. Because M is misreport-proof, we know that agent j is best-off making this report truthfully, i.e., for any possible misreport manipulation that would lead to subjective work graph G''_i it holds that: $\forall C_i$ with $j \in C_i : S_j^M(G'_i, C_i) \geq S_j^M(G''_i, C_i)$. Now, analogously to the argument in the previous proof, because M satisfies the independence assumption and is symmetric, we know that agent j also does not decrease its own score (when it is inside the choice set) when j reports to i that sybil s_j has performed c units of work for j, i.e., $w_i^{'j}(s_j, j) = c$. But we have already shown in the previous proof, that due to the single-report responsiveness property, this report leads to a score of $S_{s_j}^M(G''_i, C_i) > S_0^M$, providing sybil agent s_j with a positive score, while non-decreasing the score for agent j (whenever j is inside the choice set). This constitutes beneficial sybil manipulation number (3), which completes the proof. □

The only property that we can reasonably relax for the design of useful accounting mechanisms is the single-report responsiveness property. We can conceive of mechanisms that require two, or more generally, k positive reports about an agent, before the mechanism assigns a score strictly larger than S_0^M to that agent. If this is the case, then the simple sybil manipulations we have used for the proofs of the above theorems don't work anymore. For a successful sybil manipulation against such mechanisms, the sybil agents themselves would have to become active, and perform some work in the network, such that at least another agent (or more generally, $k-1$ other agents) makes a positive report about the sybil agent. While this would lead to some notion of k-sybilproofness, these mechanisms would also have to ignore a much larger part of the information, because in a distributed system, the probability of receiving two, or more generally, k reports about the same agent decreases exponentially. Thus, in practice, there would be a trade-off between robustness against sybil manipulations on the one side, and informativeness on the other side. Analyzing this trade-off in more detail (analytically and experimentally) is subject to future work.

The importance of robustness against sybil attacks also depends on the domain, i.e., in particular how difficult it is for the agents to perform a sybil manipulation. In P2P networks, for example, misreport attacks are much easier to execute than sybil attacks and also more beneficial. 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 of P2P file-sharing networks, protecting against misreport attacks is generally of higher importance than protecting against sybil attacks.

4.3 The Role of the Max-Flow Algorithm

We will now briefly discuss the motivation for using the max-flow algorithm inside BarterCast and BarterCast+DropEdge to compute agents' scores. Ideally, we would like to do accounting via taking the total sum of work performed and subtracting the total sum of work consumed for each agent. By using the max-flow algorithm, we essentially do a form of "bounded addition" which obviously distorts the true net work measure. The BarterCast+DropEdge mechanism would also be misreport-proof without the use of the max-flow algorithm, however, there are multiple reasons for using max-flow. First, it is useful to protect against Byzantine agents, i.e., agents that are not necessarily acting rationally but who simply try to harm the network or specific agents in the network. If such a Byzantine agent reports that agent i has consumed 1,000,000 units of work from him, and if other agents believe this report, then agent i will be unable to receive any work from those agents in the future and there is nothing i could do about it. Using max-flow bounds the influence of any agent by

the amount of work it has performed/consumed in the network, and thus makes Byzantine attacks much more difficult and costly for the attacking agent, thereby effectively preventing them in practice.

A second reason for using max-flow is to protect against sybil and collusion attacks. As we have discussed in the last section, a fully sibyl-proof accounting mechanism would have to discard too much information, and thus in practice, non-sybil-proof accounting mechanisms might be used. However, by using the max-flow algorithm, we bound the influence of any agent, thereby limiting the power of sybil attacks, making them more costly and thus less attractive for the attacking agent. Note that instead of using max-flow, we could also use other graph-based algorithms. The algorithm only needs to have two properties: first, it needs to have the "bounding property" to limit the influence of any agent proportionally to how much that agent has contributed to the system so far. Second, the algorithm must have the "transitive-trust" property, i.e., when agent i has performed some work for j and j has performed some work for k, then i should also trust agent k to some degree. We refer the interested reader to [23] where we review the most important transitive-trust mechanisms (shortest-path, EigenTrust, PageRank, Max-Flow, Hitting-Time) and introduce a new class of hybrid transitive-trust mechanisms, which could also be used for the design of accounting mechanisms.

5 Experimental Analysis: Discrete Simulations

In this section, we evaluate the mechanisms empirically via simulations to better understand the trade-offs that are made in the Drop-Edge and BarterCast mechanisms. We begin our analysis with a discrete, round-based simulation where agents perform discrete units of work in each round. In Section 6 we move on to a more realistic simulation, using the accounting mechanisms as an overlay protocol for a BitTorrent client.

5.1 Experiment Set-up

Here, 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).

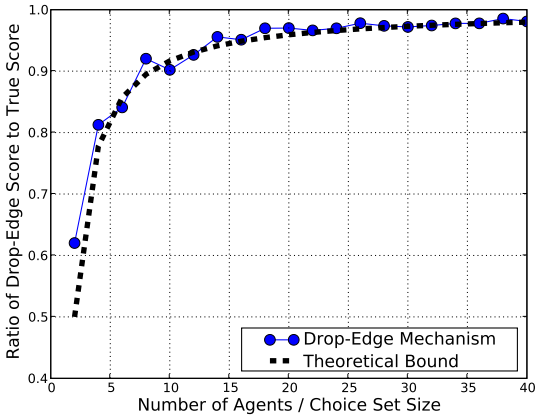


Figure 6: The approximation ratio of the scores.

5.2 Information Loss of Drop-Edge

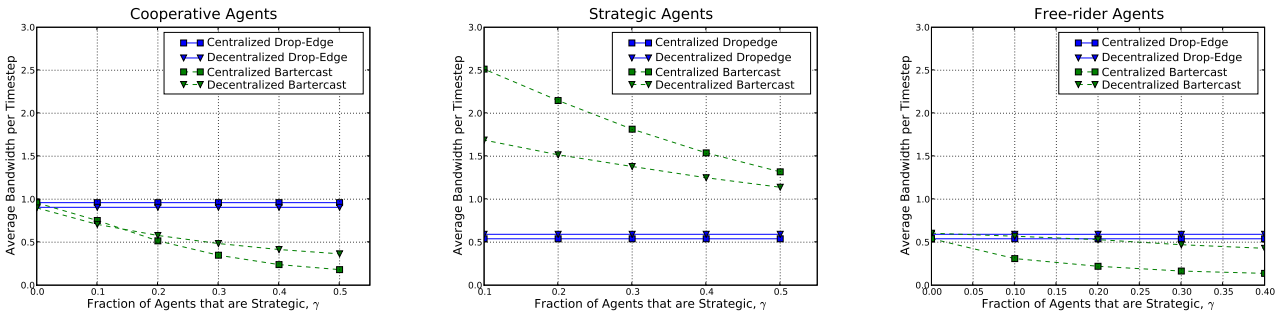
We first verify our theoretical results on information loss from Section 4.1. Fixing a choice set size $m = 5$, free-rider agent fraction $\beta = 0.5$, and strategic agent fraction $\gamma = 0.2$, we simulate networks of size $n = 10, 20, ...200$. After 100 time steps, for every agent we randomly choose 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 (see Figure 6).

5.3 Efficiency Results

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 7 (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 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 7 (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 7(b), we clearly see that strategic agents are able to sharply increase their performance compared to the other agents (see Figures 7(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



(a) (b) (c)

Figure 7: Bandwidth of different types of agents, using Bartercast and Drop-Edge. The fraction of free-riding agents is $\beta = 0.5$.

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.

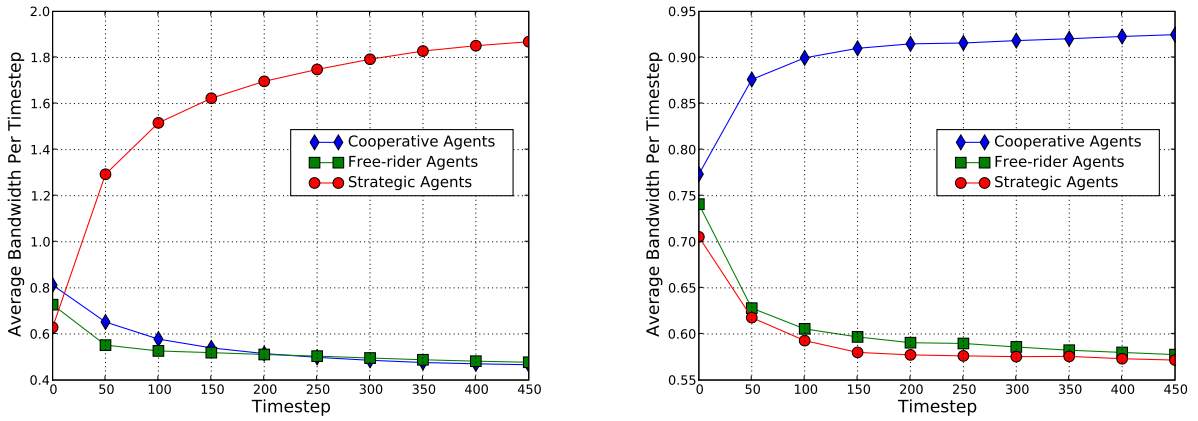
In practice, strategic misreports may also occur under Drop-Edge even though such behavior is not rational for an individual agent. We have tested Drop-Edge in settings with strategic agents (not plotted) and although the efficiency of the cooperative agents decreases slightly as the proportion of strategic 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 8 (a), we see that the benefit that strategic agents gain from misreporting in BarterCast gets even larger over time. Compare this against Figure 8 (b), 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).

6 Experimental Analysis: A BitTorrent Overlay Protocol

In this section we discuss our application of BarterCast and BarterCast+DropEdge in BitTorrent, a popular P2P file-sharing protocol. BitTorrent currently only implements a tit-for-tat policy which gives downloaders in a single swarm an incentive to upload to each other. However, there is no incentive to continue sharing the file after the download has finished. Ironically, it is even disadvantageous to share, since the consumed upload bandwidth cannot be used to do tit-for-tat in other downloads, which makes these downloads slower. Using accounting mechanisms on top of the existing BitTorrent protocol, we want to remove this incentive problem to increase the overall efficiency of the system.

The normal BitTorrent protocol maintains a limited number of simultaneous upload slots (usually 4-7 depending on the implementation). Peers that do not yet have the complete file (*leechers*), assign their slots to those peers that currently provide the highest upload rate in return, determined periodically. Peers that have the complete file (*seeders*) assign their upload slots to those peers that



(a) BarterCast (b) Drop-Edge

Figure 8: Bandwidth in BarterCast and BarterCast+Drop-Edge over Time.

have the highest download rate. Peers that get a slot are called *unchoked*, while the other peers are *choked*. Furthermore, there is one extra slot for *optimistic unchoking*. This slot is assigned via a 30 seconds round-robin shift over all the interested peers regardless of their upload rate. The protocol therefore creates a tit-for-tat data exchange based on the short-term behaviour of a peer (i.e., the bandwidth it provides in return). Due to optimistic unchoking, new peers have a chance to obtain their first pieces of data and bootstrap the process.

6.1 Accounting Mechanisms in BitTorrent

When an accounting mechanism is available in a P2P file-sharing system such as BitTorrent, this raises the question which allocation policy to use. A natural candidate would be the ranking policy which always gives preference to agents with a higher score. As we have shown in Section 5, in the discrete simulations the ranking policy works very well, and successfully separates sharers from free-riders. However, the situation is more complicated when we are targeting a system like BitTorrent that is already deployed in practice. First of all, a new BitTorrent client should be backwards-compatible with old BitTorrent clients that are not using the accounting mechanism. Secondly, a user who installs the new BitTorrent client with the accounting mechanism should have a performance at least as good as with the old BitTorrent client. But this puts some restrictions on what kind of allocation policies we can usefully employ.

Imagine agent i using BarterCast in a BitTorrent network with primarily "standard BitTorrent clients", i.e, those that do not use the accounting mechanism. A standard BitTorrent client allocates the optimistic unchoking slot to a random agent. Thus, if agent i uses the accounting mechanism to decide which agent to unchoke optimistically, this does not affect its performance. However, the remaining upload slots are normally allocated to those agents providing the highest download speed in return. Myopically, this optimizes the download performance for the uploading agent. Now, if agent i would allocate *all* upload slots based on the accounting mechanism, agent i's performance could degrade, because possibly the agents with the highest scores have uploaded a lot in the past, but they do not have any pieces to reciprocate in the current swarm. Thus, agent i would be worse off using BarterCast than using the standard BitTorrent client, and this must be avoided.

Consequentially, the ranking policy that we employ in our experiments with BitTorrent only uses the accounting mechanism to decide who to allocate the optimistic unchoking slot to. The second

policy we employ is the banning policy where an agent will never upload to an agent with a score below a certain threshold. The idea here is that the threshold is set low enough such that sharers will never reach a score lower than this threshold. Thus, banning an agent completely that has a score below the threshold only excludes the free-riders from the network. Note that both policies are easy to implement on top of any P2P file-sharing protocol and backwards-compatible with existing mechanisms:

- **ranking policy:** Peers assign optimistic unchoke slots to the interested peers in order of their scores. Therefore, a peer can not get an upload slot while peers with a higher scores are also interested and not yet served.
- **banning policy:** Peers do not assign any upload slots to peers that have a score which is below a certain threshold δ.

6.2 Simulation setup

We have built a simulator which incorporates all relevant aspects of BarterCast and BitTorrent. We simulate an epidemic Peer Sampling Service combined with the BarterCast protocol and the BitTorrent protocol.[6] Our BitTorrent simulator follows the protocol at the piece-level, including unchoking, optimistic unchoking, and rarest-first piece picking. We have combined all processes in simulation environment that can generate workloads based on either probabilistic request arrivals or data of traces.

In our experiments we simulate 100 peers active in 10 swarms (i.e., corresponding to 10 different files) during a simulated time of one week. Note that a peer can be active in multiple swarms simultaneously. A peer can be either a *sharer*, a *free-rider*, or a *strategic peer*. free-riders immediately leave the swarm after finishing a download, while sharers share every downloaded file for 10 hours. Strategic peers behave as free-riders, and in addition spread only forged BarterCast messages in which they report a maximum upload to others and zero download. All peers in our simulations have a 3 MBps downlink and a 512 KBps uplink, corresponding to common ADSL users. As these users have very limited uploading capacity, they are likely to economize on sharing, and are therefore the most important target of sharing-ratio enforcement in current file-sharing systems. Finally, in the BarterCast messages we have used $N_h = N_r = 10$.

Using our simulator, we performed two different types of experiments. First, we simulated the arrival process of agents in the network according to a poisson distribution. Second, we did trace-based simulations where the behavior of the agents was modelled closely after traces from a real BitTorrent tracker.

6.3 Poisson-based simulations

We implemented a random generator that for each peer generates Poisson arrivals of file request with an average of one request per day per peer. Requests are homogeneously distributed over the 10 files. In these simulations, 50% of the peers are sharers, 40% of the peers are free-riders, and 10% of the peers are strategic peers. In our experiments, we assess the evolution of the average scores of each group of agents, and compare the download performance for different policies. Furthermore,

[6] The actual implementation of the Peer Sampling Service is transparent to BarterCast, and it is sufficient here to note that distributed PSS protocols are feasible and do exist. We use the decentralized BuddyCast epidemic protocol that is already implemented and released as part of the Tribler file-sharing client [19] since Version 4.2

we evaluate the influence of the threshold under the banning policy, and evaluate a simple model for behavioral change.

6.3.1 Evolution of agents' scores

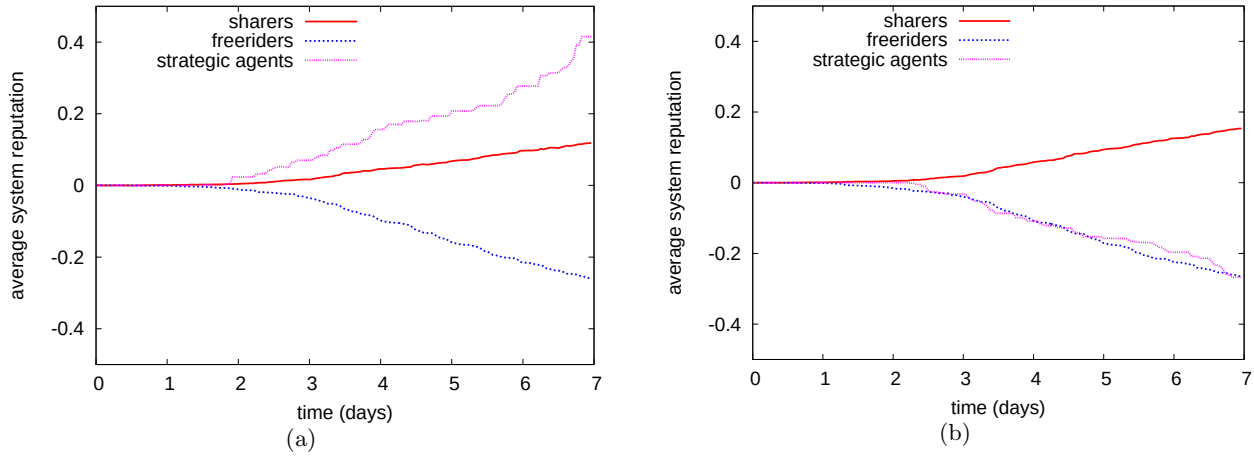


Figure 9: (a) The average system scores without Drop-Edge; (b) The average system scores with Drop-Edge.

As a first step, we evaluated how the scores of the different agents evolved over time, i.e., whether the accounting mechanism can successfully track agents' net contribution to the system. Because each agent computes a different score for each other agent in the network, we compute the *system score* $\mathbf{S}_i$ of agent i as the average of the score that each of the other $N-1$ agents assign to i:

$$\mathbf{S}_i = \frac{1}{N-1} \sum_{j \neq i} S_i(G_j, C_j) \tag{18}$$

In Figure 9 we have plotted the average scores of the sharers, the free-riders, and the strategic peers over time during the simulation. Figure 9a shows the results for the scores based on maxflow, without the use of Drop-Edge. The scores of the free-riders clearly decrease over time, while the scores of the sharers increase slightly. However, the strategic peers benefit a lot from manipulating the accounting mechanism; their scores are significantly higher than the scores of the sharers. This effect vanishes completely when the Drop-Edge strategy is applied, which is displayed in Figure 9b. Here, the strategic agents have the same average scores as the free-riders. Obviously, the Drop-Edge strategy is very successful in reducing the possibilities for individual peers to manipulate the mechanism.

6.3.2 Policy evaluation

We have compared the effect of the banning and the ranking policy on the average download performance of the peers. All results are normalized with respect to the results of simulations that do not implement any policy at all. In order to distinguish the effects of maxflow, decentralization, and Drop-Edge, we have used the following set of accounting mechanisms:

- **Central Maxflow** The actual up- and download statistics of peers are stored in a central database, instead of in local databases. The score $R_i(j)$ of peer j in the eyes of peer i is

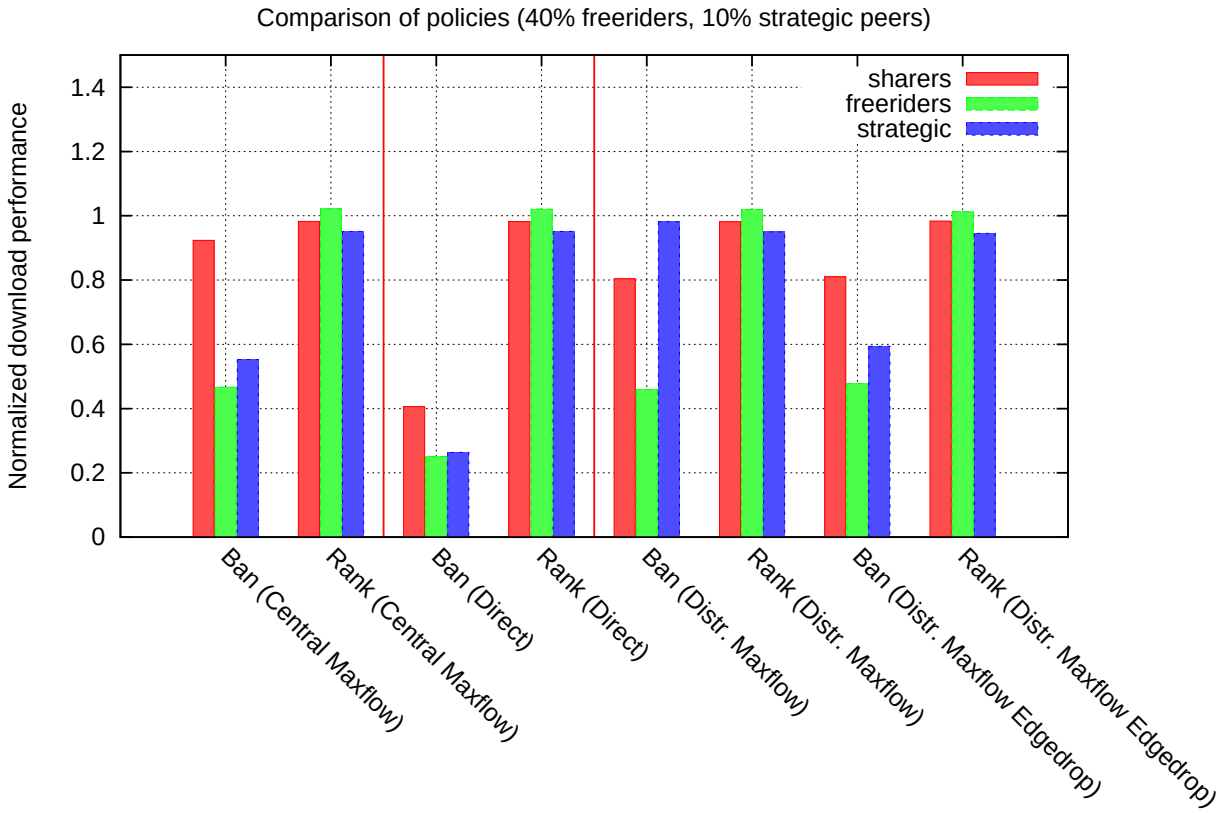


Figure 10: Comparison of the average download performance of sharers, free-riders, and strategic peers for both the banning and ranking policy with different accounting mechanisms.

computed using the maxflow algorithm and the score metric, based on the central information. Note that while the statistics are centrally stored and objective, the computed score of peer j depends on the position of peer i in the network and is therefore still subjective.

- **Direct** Instead of computing the maxflow over all paths from one peer to another peer, a peer i computes the score of peer j based on *only* the up- and download volume of direct transfers between i and j[7].
- **Distributed Maxflow** The maxflow algorithm applied by each peer based on a local database of up- and download statistics obtained by BarterCast messages.
- **Distributed Maxflow Drop-Edge** The same as Distributed Maxflow, with in addition the Drop-Edge strategy.

We have applied both the banning and the ranking policy to each of the above accounting mechanisms. For the banning policy, the default banning threshold δ is set to -0.5. In Figure 10, the normalized average download performance of the sharers, the free-riders, and the strategic peers is displayed for each of the resulting combinations of accounting mechanism and allocation policy. An

[7]Technically, this is equivalent to the maxflow algorithm with a maximum path length of 1.

immediate and very interesting observation is that regardless of how the score is computed, the ranking policy has no significant effect at all. An investigation of the results showed that this is caused by the size of the swarms. As we simulate relatively small swarms, peers have not always requests of other interested peers to fill all of their upload slots. Hence, free-riders can often find enough free slots to still have a normal performance. We argue that a policy based on ranking would only become effective if swarms are relatively large *and* peers know a significant fraction of the other peers in the network. Only then would each peer receive so many requests from other interested peers that the free-riders are outranked by the sharers.

To verify this hypothesis, we ran more experiments with varying swarm sizes. In Figure 11 we display the results from that experiment. On the x-axis we vary the accuracy of the accounting mechanism, where an accuracy of 0.4 means that, on average, the agents have accurate scores for 40% of the agents in the network, and none for the rest. First, it is easy to see that once the accuracy increases from 0% to 20%, the resulting performance doesn't change anymore. This is a positive result, because it implies that even in a large system where agents will always only have a partial view of the network, the ranking policy works well as long as each agent has some minimal amount of information about other agents. Now let's consider the effect of changing the swarm sizes. As we can see, indeed, with a very small swarm size of only 20 agents, the ranking policy is not effective in separating sharers from free-riders. However, as we increase the swarm size from 20 to 60 agents, this changes, as now the sharers have an average performance of 500Kbps and the free-riders have an average performance of 400Kbps. Thus, this verifies our previous hypothesis that the ranking policy is only effective for swarm of some minimal size.

However, Figure 11 also shows that the performance difference between sharers and free-riders does not increase further as we increase the swarm sizes from 60 to 180. It turns out, the explanation for this effect lies in the details of the BitTorrent protocol. In our simulations, each agent has five upload slots. We use the accounting mechanism to allocate the optimistic unchoking slot. The other four slots are allocated based on the standard tit-for-tat policy, and scores from past contributions in the system don't matter. Thus, sharers only benefit from the allocation of the optimistic unchoking slot, which represents one out of five slots. Thus, it is not surprising that the performance of the sharers is exactly 5 to 4 (i.e., 500Kbps to 400Kbps), the same ratio as the upload slots that are allocated to them on average. This reveals an inherent limitation of using the ranking policy in BitTorrent. Because we want to maintain backwards compatibility, and make sure that users of our mechanism are at least as well of as users of the standard BitTorrent mechanism, the maximal effect that employing the ranking policy could have is limited: we can at most reduce the performance of the free-riders by 20% compared to the sharers. In most environments, this won't be enough to incentivize the free-riders to change their behavior and become sharers. Thus, we will investigate the power of the banning policy in more detail.

The results for the banning policy in Figure 10 show several very interesting properties of the various accounting mechanisms. First of all, the banning policy applied to the centralized maxflow implementation causes the performance of both free-riders and strategic peers to be roughly cut in half. Obviously, as there are no messages involved, the strategic peers are not distinguishable from the free-riders, so the difference between those two groups is caused by random effects. When the banning policy is applied to only direct information, not only the performance of the sharers and free-riders is very low, but also that of the sharers. This can be expected; while a sharer has uploaded a lot in total, in its individual relationship with another peer it might have downloaded much more than uploaded. Hence, a significant fraction of the peers might still ban a particular sharer. Even though the free-riders and strategic peers have the lowest performance, the system as a whole suffers significantly when only direct information is taken into account. When the maxflow algorithm is

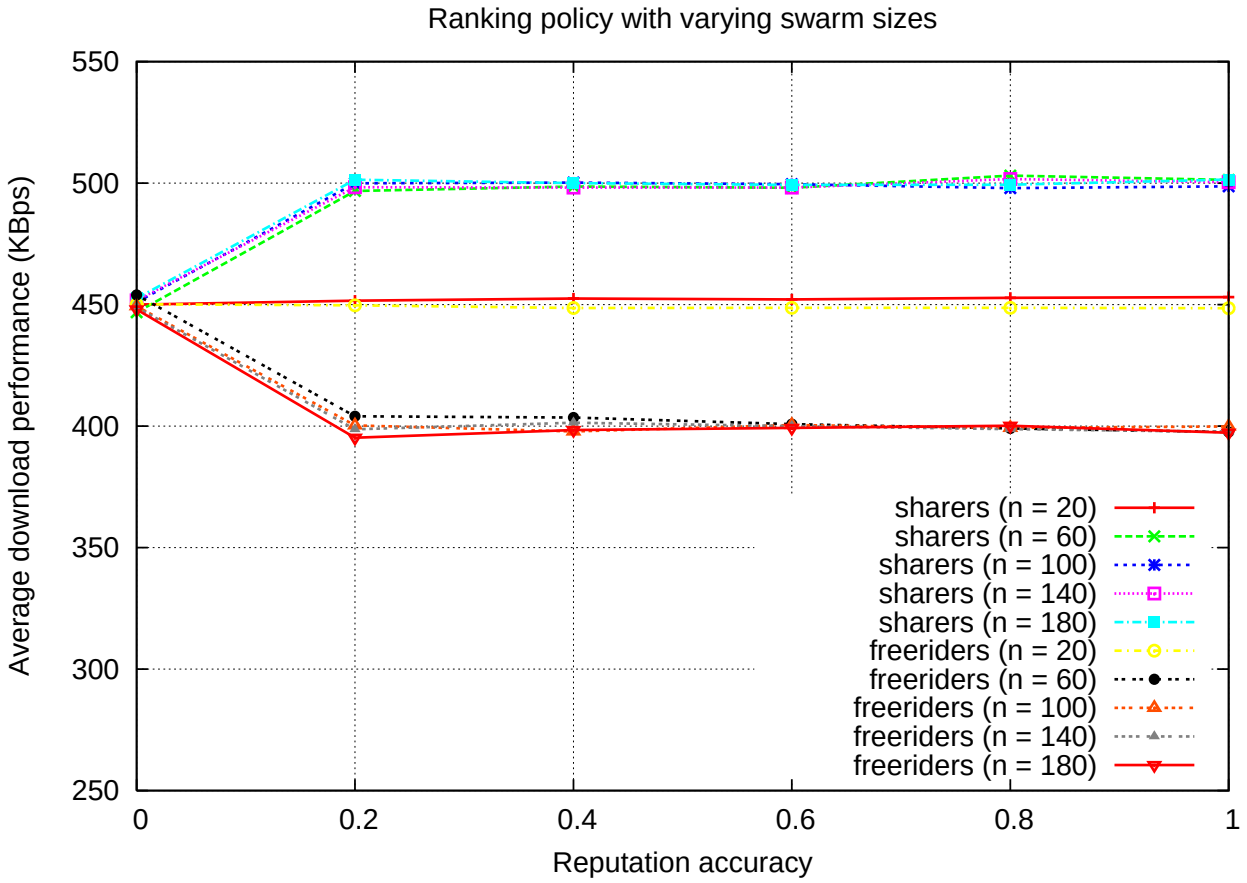
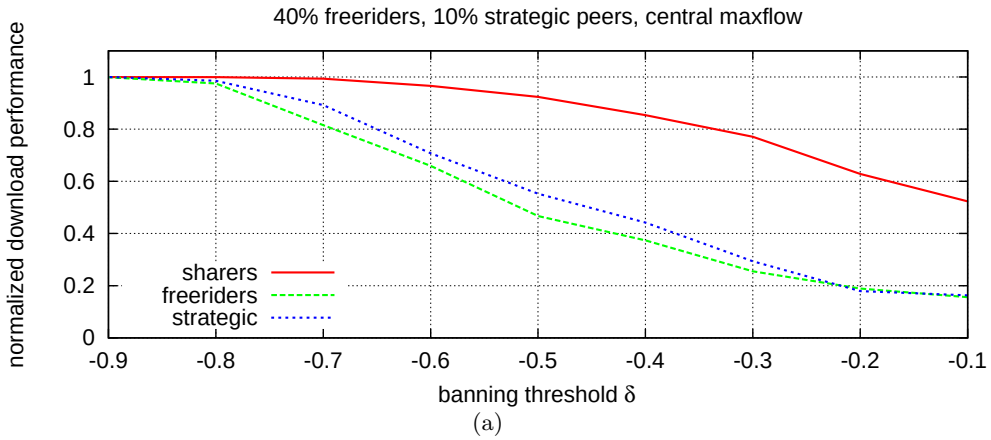


Figure 11: Performance of the Ranking Policy with different swarm sizes.

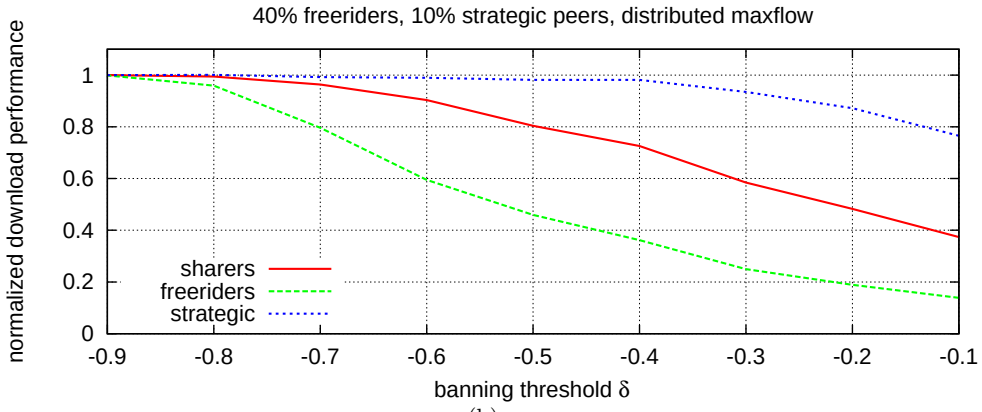
applied based on distributed information, the results are much more attractive. The penalty for free-riders is comparable to that of the central approach. The sharers also experience a somewhat lower performance, but this trade-off might pay off in the end when free-riders would become sharers because they experience a low performance when they do not (see Section 6.3.4). Finally, the effect of the Drop-Edge strategy is once again very clear: the strategic peers perform better than the sharers if Drop-Edge is not used, and they have a performance comparable to the free-riders if is used. This is consistent with the evolution of their average score shown in Figure 16.

6.3.3 Influence of the threshold

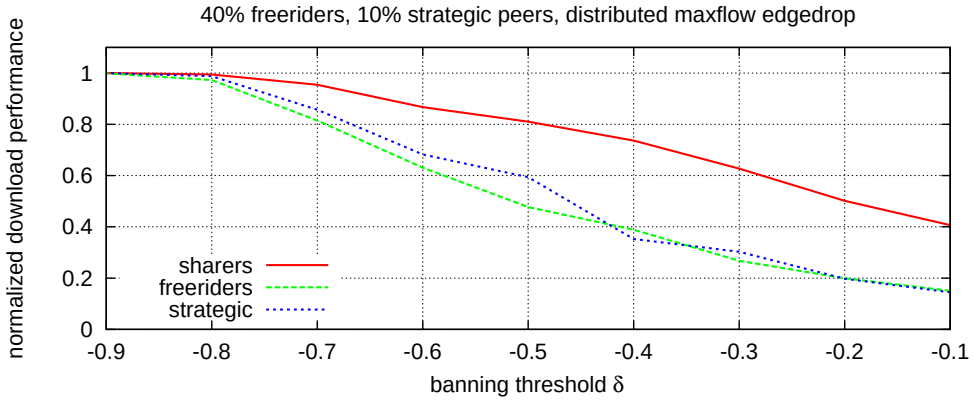
We have investigated the influence of the banning threshold on the effectiveness of the banning policy. In Figure 12, the normalized download performance for sharers, free-riders, and strategic peers is plotted for various thresholds using the Central Maxflow, Distributed Maxflow, and Distributed Maxflow Drop-Edge implementations. In Figure 13, we have displayed the same results for the free-riders and strategic peers, relative to the download performance of the sharers. The latter figure shows clearly that the more strict the threshold (i.e., closer to 0), the more severe the relative penalty of the free-riders. Strategic peers have no benefit when the Drop-Edge strategy is used. While from this perspective, the strictest threshold seems to be the best option, Figure 12 makes clear that a stricter threshold also gives a greater loss of performance for the sharers. There is no clear optimum;



(a)

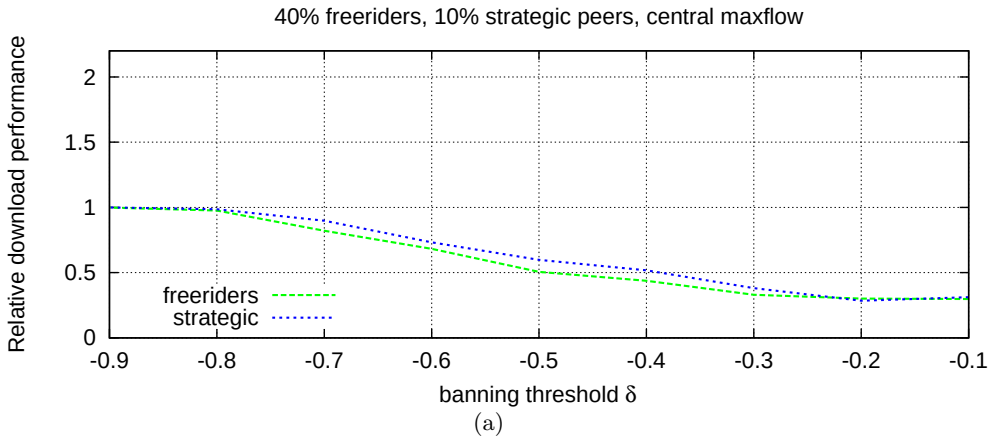


(b)

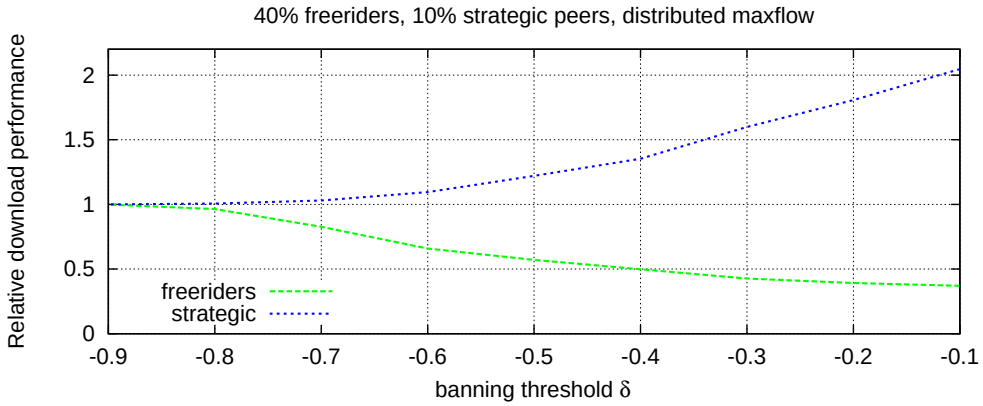


(c)

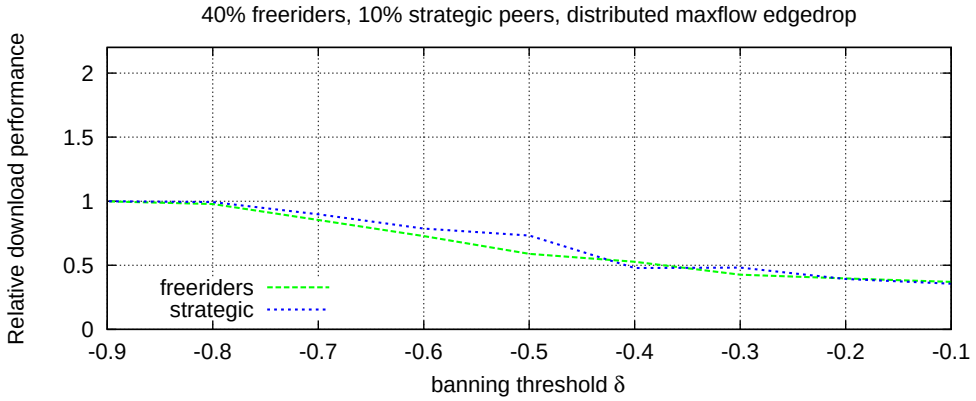
Figure 12: The normalized download performance of sharers, free-riders, and strategic peers using the banning policy for various thresholds δ under (a) Central Maxflow, (b) Distributed Maxflow, and (c) Distributed Maxflow Drop-Edge.



(a)



(b)



(c)

Figure 13: The download performance of free-riders and strategic peers *relative* to the download performance of sharers using the banning policy for various thresholds δ under (a) Central Maxflow, (b) Distributed Maxflow, and (c) Distributed Maxflow Drop-Edge.

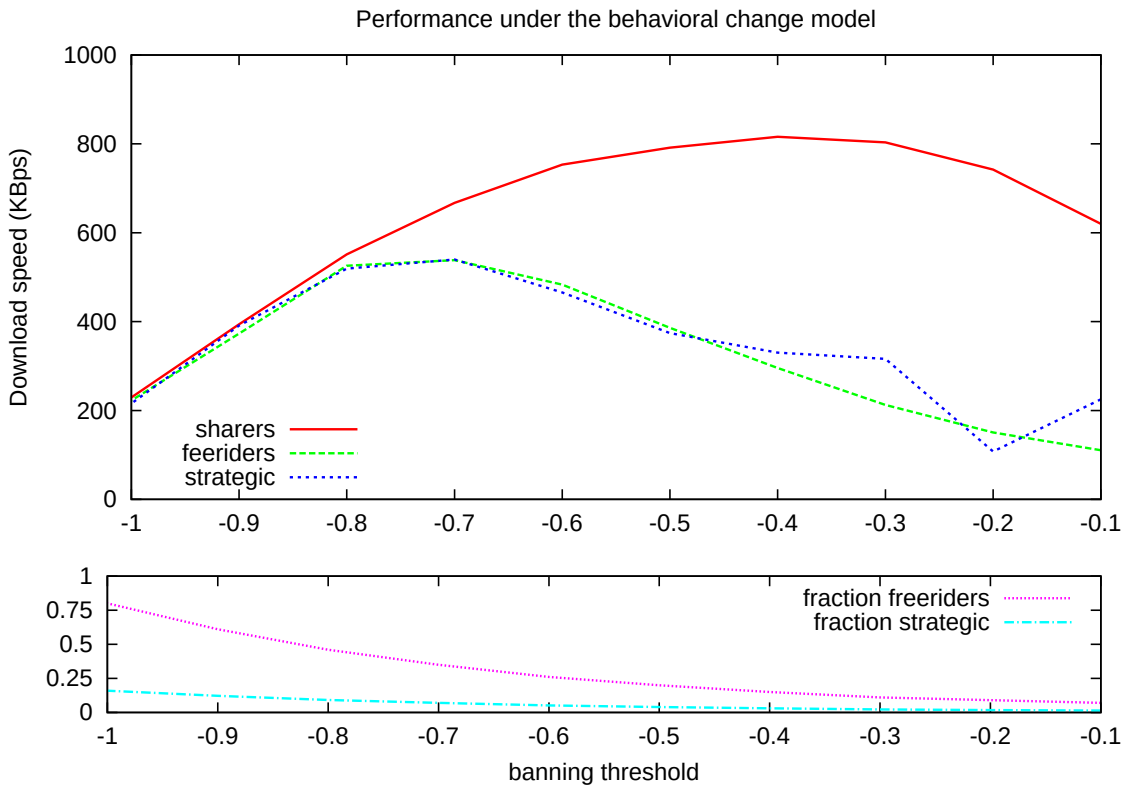


Figure 14:

system designers can consider which of the two effects is the most important. If one assumes that many free-riders will convert when they experience a severe penalty, a strict threshold is best, since in the end the overall system performance will improve for all peers because of the added resources of the 'ex-free-riders'. However, if free-rider conversion is slow, the prolonged loss of performance of the sharers might be unacceptable, and a milder threshold should be considered. In the next section, we will investigate this trade-off in more detail.

6.3.4 Behavioral change

The goal of reducing the performance of free-riders is to motivate them to become sharers. It is reasonable to expect that in practice a system with a stronger penalty for freeriding has less free-riders. The previous sections have shown that under the banning policy the penalty for free-riders gets stronger when the policy is stricter, i.e., δ closer to 0. Hence, we can expect that in reality there is some relationship between the setting of the threshold and the fraction of free-riders in the system. In order to gain an idea of the influence of this relationship on the performance of sharers, free-riders, and strategic peers, we performed simulations assuming the following behavioral relationship between the banning thresh-hold δ and the percentage of free-riders in the system$f$:

$$f(\delta) = 0.8 \cdot (\frac{1}{16})^{\delta+1}. \tag{19}$$

With the above relationship, a system with no banning (i.e., $\delta = -1$) has 80% free-riders, while a system with very strict banning (i.e., $\delta = 0$) has only 5% free-riders. As before, we assume that

in addition $0.25 \cdot f(\delta)$ of the peers are strategic. In Figure 14, we have displayed the download speed for sharers, free-riders, and strategic peers in a system where the above relationship holds. We observe that when the policy is very loose (δ close to -1), all peers have a relatively low performance. This is intuitive, since because there is hardly any penalty for free-riders, many peers will freeride, which implies only little supply of resources. As the policy gets stricter, the performance of all peers naturally starts to increase. At some point ($\delta > -0.8$), there are apparently enough sharers in the system for the banning of free-riders to become effective. Around $\delta = -0.4$ we observe that the download speed of the sharers peaks, while the penalty for freeriding is very strong. As the policy gets even stricter ($\delta > -0.4$), we see that the banning of peers starts to have a negative effect on the performance of sharers. The explanation for this effect is that even a sharer might sometimes be banned because of the use of the distributed max-flow algorithm, i.e., some agents might falsly classify a sharer as a free-rider. Thus, the trade-off between sufficient banning of free-riders versus reducing unnecessary loss of performance for sharers is clearly visible in this figure. In practice, depending on the actual relationship f, it is up to community managers and system designers to devise policies that successfully balance this trade-off.

6.3.5 The Effect of Accounting Accuracy and Noise

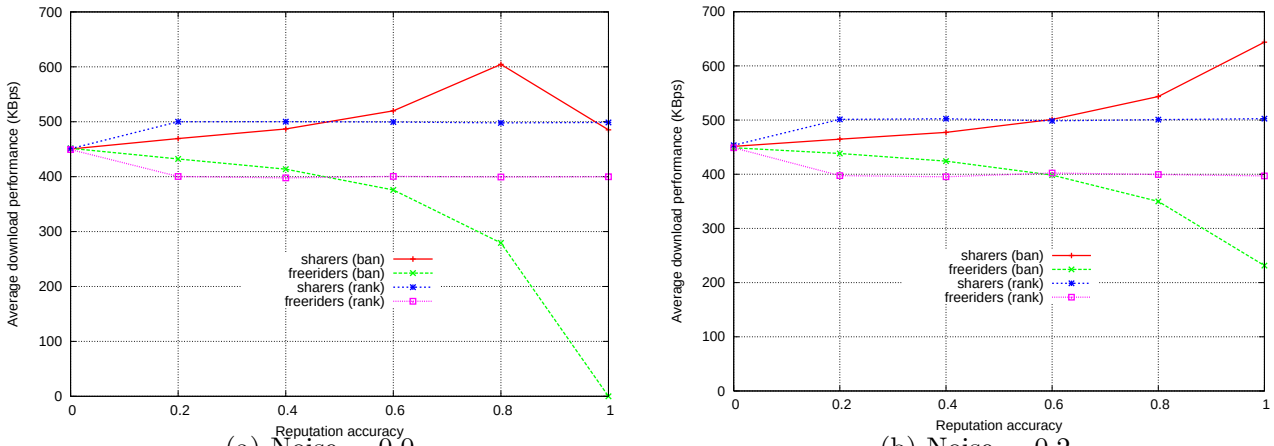
While the network size in our simulation is relatively small (100 agents), real BitTorrent networks are much larger, on the order of millions of agents.[8] In this section we study the effects of possibly disconnected graphs of agents and noisy information. In the following experiment, the agents do not compute the scores of the other agents explicitly, but instead we inject the scores, giving us the ability to control precisely how much and which information each agent has.

Fortunately, Piatek et al. [17] have shown that even in large networks, most agents are connected to each other via paths of length 2. However, not all agents will be connected to every other agent via such short paths. Given that in our max-flow implementation we only consider paths of length 3 or less, some agents will effectively have no information about some other agents in the networks. For the experiments we let the *accounting accuracy* denote the average percentage of agents that an agent has any information about. For example, an accuracy of 0.8 implies that on average, an agent is connected to 80% of the agents via paths of length 3 or less. Another effect that using max-flow causes is that the scores compute by max-flow are only approximations for the net work performed by an agent. As we have shown in Section 4.1, on average, using max-flow will lead to the same relative scores of the agents. However, for individual agents' scores, using max-flow introduces some noise. In our experiments, the *noise* corresponds to the variance of the distribution from which we draw an agent's view of another agent's score.

In Figure 15 we shown the results from these experiments. On the x-axis of all graphs, we vary the accounting accuracy between 0 (no information about any agent) to 1 (perfect information). The four graphs (a)-(d) correspond to noise values of 0.0 (no noise, i.e., perfect information), 0.2, 0.4, and 0.8. The mean of the distributions from which we draw the agent's scores were always the true scores. We also experimented with shifting the mean up or down (i.e., introducing systematic biases), but this did not lead to qualitatively different results.

The results for the ranking policy are very straightforward. Once the accuracy reaches the level of 0.2, the mechanism successfully separates sharers from free-riders, giving them a performance ration of 5 to 4, and this stays the same even as we increase the accuracy to 1.0. We have already explained

[8] We cannot simulate significantly larger BitTorrent networks on the protocol level because we need to compute each agent's score from each other agent's perspective, and this simply takes too long once we go beyond a certain network size.



(a) Noise = 0.0

(b) Noise = 0.2

(c) Noise = 0.8

(d) Noise = 2.0

Figure 15: Analyzing the effect of accounting accuracy and noise.

in Section 6.3.2 what the origin of this effect is. By comparing graphs (a) through (d) we also see that introducing noise into the system has no effect on the performance of the ranking policy, even up to a noise level of 0.8 (Figure 15 (c). The explanation is simple: the average score of the sharers is still much higher than the average score of the free-riders. Thus, even with some noise in the system, it is still very likely that the highest ranked agent (out of all agents) in any given choice set will be a sharer and not a free-rider. Only once we increase the noise to 2.0 does the result change a little bit, i.e., the performance for sharers decreases slightly and the performance of the free-riders increases slightly. However, such high values of noise are not realistic.

Now we turn our attention to the banning policy, and here the effects of accounting accuracy and noise are much more pronounced. As we see in all graphs of Figure 15, the performance difference between sharers and free-riders keeps increasing as we increase the accounting accuracy from 0 to 1. This is in fact expected for the banning policy, because every agent that another agent has no information about will be treated as an agent with a default score, and thus will not be banned. Finally, let's consider the effect of adding noise when using the banning policy. Here we see the biggest effects: even going from no noise to a noise level of 0.2, the performance difference between sharers and free-riders decreases significantly. For noise level 0.8, the performance difference achieved via the banning policy is almost consistently smaller than the one achieved via the ranking policy, except for very high accuracy values, where the two policies perform essentially equally well. Again, this behavior is expected. By adding noise to the system, not only are some of the free-riders not banned that should be banned, but also some of the sharers are now banned that should not be banned. And the frequency of these mistakes is much higher than for the ranking policy, because it applies to every agent that is considered for any upload slot, not just the agents being considered for the optimistic unchoking slot.

Now, to conclude this analysis we have to make some estimates regarding what accuracy and noise levels we can expect in real BitTorrent systems. As we have shown in our experiments before, even using the max-flow algorithm, the mechanisms compute scores that are highly correlated with agent's true net work. Thus, in practice, if an agent has information about another agent, the noise level can be expected to be relatively low. Furthermore, based on the results by [17], and assuming that we use a max-flow algorithm with a maximum path length of 3, an accounting accuracy of 0.99 can be expected in real BitTorrent networks. Thus, we should consider Figure 15 (b) and look towards the right end of the graph where we see that the banning policy causes a very large performance difference for sharers and free-riders, which is the effect we desired. Note that all of the results in Figure 15 are for one fixed value for the banning threshold. As we have explained in the previous section, by fine-tuning the banning threshold, we can potentially further increase the performance difference between sharers and free-riders.

6.4 Trace-based simulations

We also performed simulations based on network traces from the popular private BitTorrent tracker `filelist.org`. The traces contain detailed behavior of all peers that were active in the file-sharing network, including uptimes, downtimes, connectability, and file-requests. We have chosen this tracker because it is not possible to scrape such detailed information from most other trackers. The file sizes in the trace range from several tens of megabytes to about one to two gigabytes, representing mostly audio and movie files. As the traces do not contain information about the precise download process and finish time of downloads, we could use only the starting time in our simulations. We randomly divided the peers into 50% sharers and 50% free-riders, and simulated the behavior described above. In our experiments, we compare the computed score of the peers with their real net contribution.

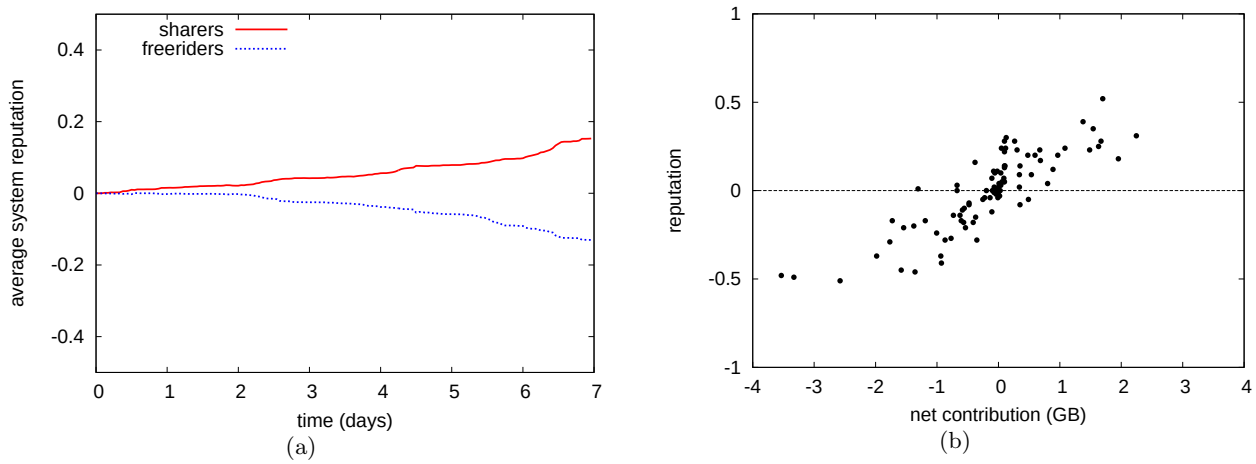


Figure 16: (a) The system score in the system for both sharers and free-riders; (b) A scatter plot of the system score versus the net contribution of peers (each dot represents one peer).

Also, we compare the banning and the ranking policy in terms of download performance.

6.4.1 Contribution versus score

We compare the real behavior of a peer with its average score in the system. In order to measure a peer's real behavior, we compute a peer's *net contribution* which is defined as its real total upload minus its real total download during the simulation period.

Figure 16 (a) shows a plot of the average system score of both the free-riders and the sharers throughout the simulation period of one week. It is visible that the scores quickly diverge and that free-riders are clearly distinguished from sharers. In Figure 16 (b) the correlation between the net contribution and the system score of individual peers is plotted, and the strongly positive correlation is apparent. Thus, despite the fact that peers have incomplete information about other peers, a peer's resulting score in the system is a good representation of its real behavior.

6.4.2 Results for the banning policy

To validate our results from the Poisson-based simulations, we have assessed the banning policy using trace-based simulations. Figure 17 shows the average download speed of the sharers and the free-riders throughout the simulation period. First of all, we observe that during the first days the average speed of free-riders is higher than that of sharers. This is because free-riders do not allocate any upload bandwidth to the sharing of complete files and therefore have more upload capacity available to do tit-for-tat in their downloads. After some time however, the speed of the sharers increases and the speed of the free-riders decreases under both policies, leading to a significantly higher speed for the sharers. At the end of the simulation period, the free-riders have only about 50% of the speed of the sharers if the banning policy is used. Thus, we replicated the results from the Poisson-based simulations, showing that the accounting mechanism successfully separates sharers from free-riders.

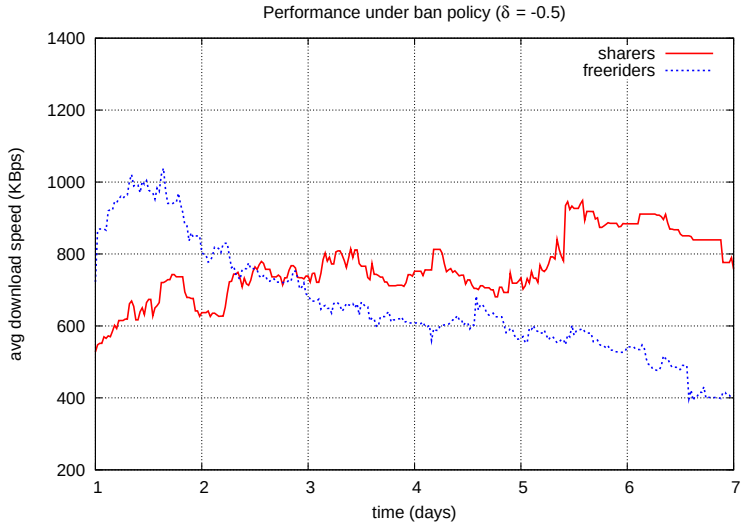


Figure 17: The average download speed using the banning policy with δ equal to -0.5.

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 extension 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. However, we have also shown that neither BarterCast nor BarterCast+Drop-Edge is robust against sybil manipulations. This is no coincidence, as we have shown that there exists no sybil-proof accounting mechanism that is responsive even to single agents' reports. We exaplain that the design of sybil-proof accounting mechanisms would require relaxing this requirement and that sybil-proof mechanisms would have to discard a significant amount of information. Thus, in practice, a system designer would have to decide how important sybil-proofness is on the one hand vs. informativeness of the accounting mechanism on the other hand.

Via discrete simulations, where agents perform discrete units of work in each round, we have shown that the good theoretical approximation ratio of the BarterCast+Drop-Edge mechanism also translates into good system efficiency. When strategic agents are present, the Drop-Edge Mechanism clearly outperforms BarterCast-basic: cooperative agents have higher efficiency, while free-riding agents have lower efficiency and we have shown that the magnitude of this effect even grows over time. To test the effectiveness of accounting mechanisms under more realistic conditions, we have implemented various accounting mechanisms into Tribler, a real P2P file-sharing client that is already deployed and being used by thousands of users. Using Tribler, we were able to run simulations at the BitTorrent protocol level and thus obtain much more realistic experimental results. First, and most importantly, we have found that the requirement of being backwards-compatible with existing BitTorrent clients puts some additional constraints on which accounting mechanisms we can usefully employ on top of a BitTorrent protocol. We have discussed in detail that using the ranking policy to make allocation decisions has limited power in BitTorrent, because it can at most achieve at 5 to 4 ratio regarding the performance of sharers vs. free-riders. The banning policy, in con-

trast, is more powerful, and with a correctly-tuned banning threshold, it can separate sharers from free-riders such that sharers' performance is more than twice as high as that of free-riders. Under such conditions, it is realistic to assume that a significant fraction of free-riders would change their behavior and become sharers. Assuming such a behavioral change, we were able to show that using BarterCast+Drop-Edge we can achieve system efficiencies significantly higher than with the standard BitTorrent protocol. We have also provided a detailed analysis of the effects of accounting accuracy and noise on the accounting mechanisms. It is necessary that the accounting accuracy is relatively high and noise levels are relatively low for the banning policy to separate the performance of sharers and free-riders to a large enough degree. Fortunately, based on previous results and our own experiments, we concluded that in real P2P file-sharing networks, we can expect relatively high accounting accuracy and low enough noise levels, such that accounting mechanisms can be expected to be successful, even in very large networks with millions of agents.

References

[1] E. Adar and B.A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10):2, 2000.

[2] Noga Alon, Felix Fischer, Ariel D. Procaccia, and Moshe Tennenholtz. Sum of us: Strategyproof selection from the selectors. In *Working Paper*, 2010.

[3] Nazareno Andrade, Miranda Mowbray, Aliandro Lima, Gustavo Wagner, and Matei Ripeanu. Influences on cooperation in bittorrent communities. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 111–115, New York, NY, USA, 2005. ACM. ISBN 1-59593-026-4. doi: http://doi.acm.org/10.1145/1080192.1080198.

[4] Tina Balke, Marina De Vos, Julian Padget, and Frank Fitzek. Using a normative framework to explore the prototyping of wireless grids. In *Proceedings of the Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems at AAMAS*, 2010.

[5] Alice Cheng and Eric Friedman. Manipulability of PageRank under Sybil Strategies. In *Proceedings of hte First Workshop of Networked Systems (NetEcon06)*, 2006.

[6] Bram Cohen. Incentives build robustness in bittorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, Berkeley, CA, June 2003.

[7] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of Electronic Commerce (EC)*, 2004.

[8] Michal Feldman, Christos Papadimitriou, John Chuang, and Ion Stoica. Free-Riding and Whitewashing in Peer-to-Peer Systems. *IEEE Journal on Selected Areas in Communications*, 24:5: 1010–1019, May 2006.

[9] Eric Friedman, Paul Resnick, and Rahul Sami. *Algorithmic Game Theory*, chapter Manipulation-Resistant Reputation Systems, pages 677–698. Cambridge University Press, 2007.

[10] Eric J. Friedman, Joseph Y. Halpern, and Ian 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.

[11] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *Proc. of NOSSDAV'03*, June 2003.

[12] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Twelfth International World Wide Web Conference*, pages 640–651, 2003.

[13] R. Krishnan, M. Smith, Z. Tang, and R. Telang. The virtual commons: why free-riding can be tolerated in peer-to-peer networks. In *Workshop on Information Systems and Economics*, December 2003.

[14] Qiao Lian, Yu Peng, Mao Yang, Zheng Zhang, Yafei Dai, and Xiaoming Li. Robust incentives via multi-level tit-for-tat. In *Proc. 5th International Workshop on Peer-to-Peer systems (IPTPS)*, 2006.

[15] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in bittorrent is cheap. In *Proc. of HotNets-V*, 2006.

[16] Michel Meulpolder, Johan Pouwelse, Dick Epema, and Henk 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.

[17] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas 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.

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

[19] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. van Steen, and H.J. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 19, 2008.

[20] Paul Resnick and Rahul Sami. Sybilproof transitive trust protocols. In *Proceedings of Electronic Commerce (EC)*, 2009.

[21] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[22] Sven Seuken, Jie Tang, and David C. Parkes. Accounting mechanisms for distributed work systems. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, Atlanta, GA, July 2010.

[23] Jie Tang, Sven Seuken, and David C. Parkes. Hybrid transitive trust mechanisms. In *Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems*, Toronto, CA, May 2010.

[24] V. Vishnumurthy, S. Chandrakumar, and E.G. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.