Work Accounting Mechanisms: Theory and Practice*

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

Working Paper

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 even in casual car-pooling communities. 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 first describe BarterCast, a fully decentralized information exchange system, where individual agents send and receive reports about the work they have performed/received. We show that a straw man solution is highly susceptible to misreport manipulations. Next, we introduce the Drop-Edge mechanism 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 our mechanisms, we first present results from a discrete, round-based simulation, showing that BarterCast-Drop-Edge 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 Mechanisms, Peer-to-peer.

*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 [31], [22].

†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 like BitTorrent, where users upload and download files to and from other users, or in ad-hoc wireless networks where individual peers route data packages for each other. 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, the long-term viability of distributed work systems relies on roughly balanced work contributions. Otherwise, strategic agents may seek to free-ride on the system, i.e., minimize the work they perform for others and maximize the work they consume.

Current systems, including BitTorrent, 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. This "local balance" 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. Pouwelse et al. [26] found that this incentive problem has significant effects in practice as more than 80% of BitTorrent users go offline immediately once they have finished downloading. *Accounting mechanisms* solve this problem by keeping long-term tallies of work performed and consumed by each user. This gives users an incentive to share even after they have finished downloading, and thus increases system efficiency.

A particular challenge 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 agents perform small units of work for each other (e.g., transmitting a few bytes), for limited periods of time (e.g., a few seconds or minutes), where no contract covers the interactions, and where no real or virtual currency can be used, because the institutional requirements for the exchange of payments are not available. Furthermore, we assume that there is no a priori trust relationship between agents, and that an agent can only earn trust by performing work. Because accounting mechanisms rely on voluntary reports, however, a major challenge is to provide robustness against strategic manipulations. The two manipulations we consider are *misreports*, where an agent overstates the amount of work contributed or consumed, and *sybil manipulations*, where an agent creates *fake* sybils (or copies of itself). We design accounting mechanisms that are *incentive-compatible*, in the sense that no agent has an incentive to manipulate the mechanism.

1.1 Accounting vs. Reputation Mechanisms

Misreport and sybil manipulations are well-studied in the related literature on trust and reputation mechanisms [11]. However, the results from this literature do not translate to accounting mechanisms. First, 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. Second, sybil manipulations are much more powerful against accounting mechanisms. For a search engine, the primary concern is that an agent could increase the reputation of its website by creating a set of sybils that are linking to the original website, but an agent does not care about the reputation of the sybils themselves. In a distributed work system, in contrast, if an agent can create sybils with a positive score, then these sybils can receive work from other users without negatively affecting the score of the original agent. While various reputation mechanisms have been proposed that are sybil-proof (e.g., maxflow, hitting-time [6, 32]), these results do not translate to accounting mechanisms. Third, once an agent has a high reputation it can benefit from that for a long time. For example, a website with a high PageRank [24] benefits from lots of visitors without affecting its reputation. In distributed work systems, in

contrast, an agent benefits from a high score by getting priority for receiving work in the future, which in turn decreases its score again. Thus, accounting scores are inherently temporary. Finally, 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 need to be "added together", to determine the overall net contributions of that agent.

1.2 Real-World Applications for Accounting Mechanisms

There are many application domains where accounting mechanisms can help to provide proper incentives and increase overall system efficiency. For example, the performance of P2P file sharing systems crucially depends on the contribution of resources by their participating users. Free-riding is a well-known issue in P2P research; its effects have been empirically measured [1, 29] and extensively analyzed [3, 9, 18, 21]. None of the existing decentralized systems provides its users with a long-term incentive to upload data to others [22], which results in large efficiency losses in practice. Centralized systems like private BitTorrent trackers [23, 36] have managed to sustain long-term contributions from its members, even though many of them can be manipulated in various ways. However, these systems generally assume some kind of central monitoring which we do not assume, and a formal study of these systems has been missing such that the incentives at play are still unclear.

A possible future application for accounting mechanisms is content distribution in 3G networks. 3G bandwidth is a very scarce resource and downloading data via a 3G network is relatively slow and requires a lot of battery power from a smartphone or similar device. In contrast, Wifi networks are cheap and fast, and smartphones require much less battery power to connect to a Wifi network than to a 3G network. There is potential to use 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. In such situations, the efficiency can be increased if only one of the users downloads the data via the 3G, and then (automatically) distributes it via ad-hoc Wifi connections to the other users. 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 user would like to be the one who downloads and then distributes the data via 3G. A similar problem arises in an application proposed by Webb et al. [35], where multiple 3G antennas are shared to increase the download speed for an individual user. Again, with a naive implementation, it would be to a user's disadvantage to permit others to use the 3G connection on his device. Using accounting mechanisms, these incentive problems can be solved, balancing the work load over time, and giving each user an incentive to consume and perform work at different points in time.

A third potential 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 from other organizational units through your own network might reduce the amount of bandwidth you can use yourself. Again, a properly designed accounting mechanism

can address this problem by making sure that over time, each network routes approximately the same amount of data, thereby providing an incentive to collaborate.

Accounting mechanisms can also be applied in domains that involve larger units of work by *human* agents, but where formal contracts or monetary transfers are undesirable for some exogenous reason. Consider a carpooling network where drivers pick up and drop off passengers at different locations in a city. According to the website of the *Casual Carpool Sites* from the Bay Area[1], no monetary payments are made from passengers to drivers, except for shares of the tolls. While participating in this network as a passenger seems very attractive, the drivers also have some advantages, including the passengers' shares of the tolls and the right to use car-pooling lanes. Still, there seems to be more demand than supply, as is indicative by the warning on the website not to "line-jump". By using an accounting mechanism that gives priority to passengers that have themselves offered rides to others in the past, proper incentives for becoming a driver could be established.

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 used here for gossiping voluntary work reports. In Section 3 we present the first formal model for the design of incentive compatible accounting mechanisms. A strawman solution, the Basic mechanism, is susceptible to misreport manipulations, even though it is built around a max-flow algorithm, which is robust against manipulations for reputation system. We introduce the Drop-Edge mechanism, which removes any incentives for agents to misreport information, by selectively dropping information dependent on the decision context. In Section 4 we provide a theoretical analysis of accounting mechanisms, where we show that the information loss of Drop-Edge due to dropping some of the information is small and vanishes in the limit as the number of agents in the network gets large. We consider sybil attacks and prove an impossibility result, that under reasonable assumptions, no sybil-proof accounting mechanism exists. This is in stark contrast to reputation systems, where mechanisms based on max-flow have been shown to be sybil-proof. We show, however, that a weaker form of robustness, K-sybil-proofness, can be achieved for a restricted class of attacks.

In Section 5, we provide an extensive experimental evaluation of the Drop-Edge mechanism for general distribution work systems, using a discrete, round-based simulation. We show that, compared to the Basic mechanism, Drop-Edge leads to much higher performance for cooperative agents, because of its robustness against misreport attacks. 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 cooperative agents and free-riders. In Section 6 we provide results from experiments using accounting mechanisms as an overlay protocol for the BitTorrent P2P file sharing network. Using Tribler, a real file sharing client that is already deployed and being used by thousands of users, we run simulations at the BitTorrent protocol level. We consider both a ranking policy and a banning policy for making work allocation decisions in BitTorrent, based on aggregate accounting information. We show that using the ranking policy, which allocate the optimistic unchoking slot to the agents with the highest score, the power of accounting mechanisms is inherently limited in BitTorrent. However, we show that using the banning policy, which bans agents whose score is below a certain threshold, we can significantly separate the performance of cooperative agents and free-riders, likely enough to induce free-riders to become cooperative. Based on all experimental results,

[1] `http://www.ridenow.org/carpool`

we conclude that the DROP-EDGE accounting mechanism can successfully separate cooperative agents from free-riders, and assuming some kind of behavioral change (i.e., free-riders becoming cooperative over time), we can achieve a total efficiency that is higher than with the standard BitTorrent protocol.

1.4 Related Work

The design of incentive-compatible distributed work systems has been a long-standing goal of the systems and AI community [14]. In particular, since the advent of popular P2P file sharing networks like Napster, Gnutella, Kazaa, and BitTorrent, this area has attracted a lot of attention. Early on, multiple studies have shown that users in these networks cheat in various ways. Adar and Huberman [1] have shown that a majority of Gnutella users free-ride and Lian et al. [20] have shown that users of the Maze network successfully perform whitewashing and collusion attacks.

Despite the important differences between accounting mechanisms and reputation mechanisms, the related literature on transitive trust and reputation mechanisms [11] is an important precursor to our own work. Gupta et al. [15] present a reputation system that is partially distributed, but relies on the authority of a single agent that stores peer reputations. Kamvar et al. [16] present EigenTrust, an algorithm for reputation management in P2P networks that is based on distributed computations of globally consistent trust vectors. However, it relies on pre-trusted peers for convergence and its aim for global consistency assumes a rigid network of peers. Karma [34] is a system in which a distinguished set of nodes keep track of the transaction balances of peers. However, they also assume the existence of a set of pre-trusted agents that store global information about all other peers, and the discussion of incentive issues is completely omitted. Feldman et al. [9, 10] study the challenges involved in providing robust incentives against free-riding, whitewashing, and misreport attacks in P networks. They introduce a reputation mechanism based on max-flow, but implicitly assume that all nodes have a complete view of the network, which is unrealistic in large, dynamic communities. Furthermore, the mechanism they propose is not misreport-proof in our setting.

An interesting, but orthogonal direction is provided by studies of virtual currencies (in our domain, the institutional requirements for a transferable currency do not exist). Friedman et al. [12] study scrip systems that rely on a trusted and transferable currency. They show how to determine the optimal amount of currency in a system to maximize efficiency. Kash et al. [17] study the effect of hoarders and altruists on such scrip systems. Dandekar et al. [8] also study credit networks, but instead of relying on a globally-trusted currency, they employ locally-trusted IOUs. Their study focuses on questions regarding the effect of network structure on credit liquidity, and largely ignores questions regarding incentive-compatibility.

One of the largest steps forward regarding the implementation of robust incentives in a real-world P2P system used by millions of users is the BitTorrent protocol, proposed by Cohen [7]. In contrast to previous protocols like Napster or Gnutella [30], BitTorrent uses a policy with short-term, direct incentives, resembling to a large degree a simple tit-for-tat mechanism. However, while this mechanism is successful for short term transactions, BitTorrent offers no incentives for long-term sharing of content. In practice, Pouwelse et al. [26] found that a majority of BitTorrent users go offline immediately after finishing a download. An interesting idea to address this propoblem was proposed by Piatek et al. [25]. They study what one might call "decentralized accounting mechanisms" and find empirically that P2P file sharing networks demonstrate a *small-world effect*, where 99% of peers exchanged data with a common third party. They propose to use well-connected intermediaries to broker information, but without providing proper incentives to the intermediaries to behave truthfully. Our own work builds on this idea, and in particular the fully decentralized information exchange protocol BARTERCAST exploits this *connectedness* of many P2P networks. However, we additionally provide

a formal framework to study the incentive properties of such systems and propose a mechanism that is misreport-proof, the major concern in the design of accounting mechanisms.

Another important concern in the design of accounting mechanisms is *sybil-proofness*. Cheng et al. [5, 6] have studied this problem for reputation mechanisms. One of their important findings is that no globally-consistent reputation mechanism can be sybil-proof, but that *subjective* mechanisms based on max-flow algorithms can be sybil-proof. While their work influenced our thinking about sybil-proofness, unfortunately, their results do not translate to our domain, due to the differences between accounting and reputation mechanisms (especially the ability of a sybil to *receive* work). A recent paper by Resnick and Sami [28] also specifically addresses the problem of sybil-proof transitive trust mechanisms. However, in their model, the individual transactions are risky and can have a positive or negative outcome, and they focus on limiting the effect of a powerful adversary. In contrast, in our domain, the individual transactions are not risky. Instead, our focus is on computing accounting scores that are proportional to the net work contributed by the agents. Our mechanism shares some similarities with a mechanism proposed by Alon et al. [2], who consider voting environments where the set of candidates coincides with the set of voters, and our theoretical analysis regarding the information loss of DROP-EDGE was inspired by their analysis.

To the best of our knowledge, we are the first to provide a formal framework to study accounting mechanisms and their incentive properties, and to point out the important differences between reputation and accounting mechanisms. Furthermore, we are not aware of any practically feasible mechanism that is decentralized to the same degree as our proposal, misreport-proof, and tested under realistic conditions.

2 Distributed Work Systems

Consider a distributed work system of n agents (or peers) 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.*

The true work graph is unknown to indivudal agents because they 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.*

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

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

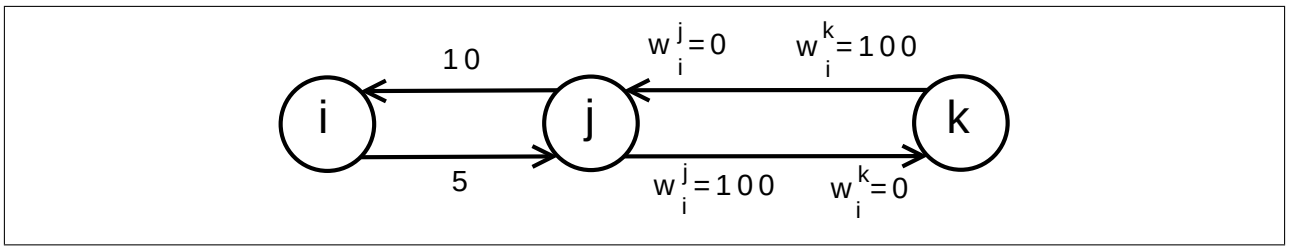


Figure 1: A subjective work graph from agent i's perspective. Edges where i has direct information have only one weight. Other edges can have two weights, corresponding to the possibly conflicting reports of the two agents involved.

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. Every agent has different *private* information about his own interactions, and the other information reported to the center is not necessarily correct. Thus, the resulting subjective work graphs for the agents can differ, no matter whether centralized or decentralized information exchange systems are used.

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*. BarterCast is such a fully decentralized, lightweight information exchange system. Each agent keeps a private history of its direct interactions with other agents and obtains information about the rest of the network by exchanging a selection of its private history with others using bilateral messages. We assume that agents can discover other agents 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, each agent will have an incomplete view and out of date view of the whole network.

3 Accounting Mechanisms

3.1 Preliminaries

In a 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

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.

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.*

The role of an accounting mechanism is to compute a *score*[2] for each agent $j \in C_i$, proportional to the net work contributed to the system, to allow agent i to differentiate between cooperative and free-riding agents. We assume that an agent has no *a priori* bias towards assisting one agent over another.

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 $S_{ij}^M(G_i, C_i)$, for any agent $j \in C_i$, as viewed by agent i.*

We let S_0^M denote the *default* score that accounting mechanism M assigns to an agent about which no information regarding work consumed or performed is available (i.e., the two agents are disconnected in the subjective work graph). Once the accounting mechanism has computed a score for each agent in the choice set, the agent uses an *allocation policy* to decide to whom to allocate work to (see Figure 2). Thus, the accounting mechanism together with the allocation policy matches work-performing agents with work-seeking agents. 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 $j \in \arg\max_{k \in C_i} S_{ik}^M(G_i, C_i)$, 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 $\delta \in \mathbb{R}$, agent i performs one unit of work for an agent chosen uniformly at random from $\{j \in C_i | S_{ij}^M(G_i, C_i) \geq \delta\}$.*

3.2 Agent Population and Strategic Manipulations

We adopt the model and terminology of Meulpolder et al.[22], and assume a population that consists of a mixture of *cooperative* agents (or sharers), who always contribute work, and *lazy free-riders* who

[2]Note that we purposefully chose to use the term "score" instead of "reputation value" even though this is in contrast to prior work by Meulpolder et al. [22]. Our goal is to clearly distinguish between accounting and reputation mechanisms and to emphasize that outputs of such mechanisms have very different meanings.

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. In Section 4.2 we study a second class of attacks on the accounting mechanism called *sybil attacks*, where an agent inserts fake agents into the network to manipulate the mechanism. Note that we model only strategic behavior with regard to manipulating the accounting mechanism and do not consider, for example, manipulations on the information exchange protocol.

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:*

- $S_{ij}^M(G_i', C_i) \leq S_{ij}^M(G_i, C_i)$, *and*
- $S_{ik}^M(G_i', C_i) \geq S_{ik}^M(G_i, C_i)\ \forall k \in C_i \setminus \{j\}$.[3]

3.3 The Basic vs. the Drop-Edge Mechanism

In this section, we first present a straw-man mechanism called the Basic mechanism, first introduced by [22]. We show that the Basic mechanism can easily be manipulated via misreports and then introduce the Drop-Edge mechanism that removes the incentive to misreport.

Definition 9. ***(Basic 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 Basic *Score of agent j as $S_{ij}^B(G_i, C_i) = MF_{G_i^B}(j,i) - MF_{G_i^B}(i,j)$.*[4]

In the Basic mechanism, an agent takes its own information over reports from others (1). Given two reports, it takes the maximum of the two (2). 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. By taking the maximum of the two reports, an agent always uses the most up-to-date information (in the case of non-strategic reports). The motivation for using the max-flow algorithm is that it 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. In Section 4.3, we discuss in more detail how this limits the power of strategic manipulations and also protects against Byzantine attacks (i.e., arbitrary attacks not necessarily originating from rational agents).

It is easy to see that the Basic mechanism can be manipulated via misreports. We illustrate two attacks in Figures 3 and 4. We always show the subjective work graph from i's perspective and the

[3] Note that the first requirement is equivalent to *value-strategyproofness* as defined for trust mechanisms, and both requirements together imply *rank-strategyproofness* [6].

[4] The specification of the Basic mechanism here differs from the one presented in Meulpolder et al. [22] 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 relative scores of the agents.

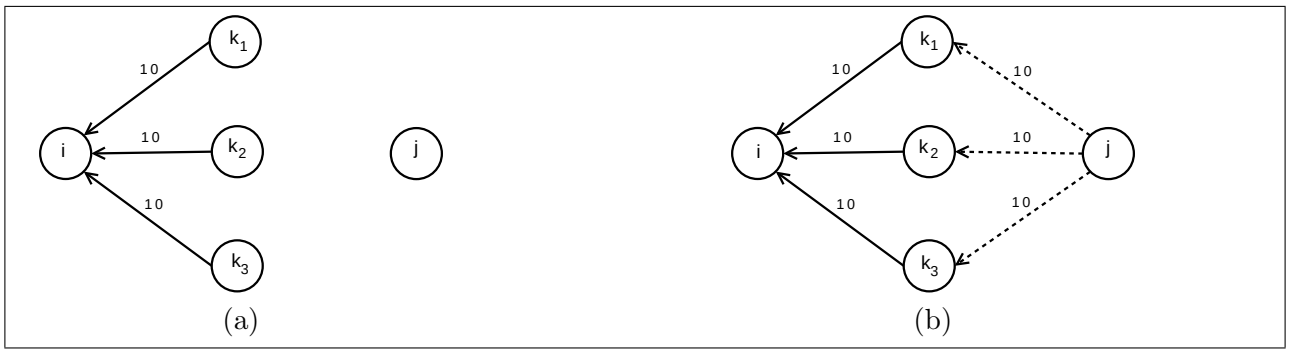


Figure 3: (a) A work graph based on true reports. (b) The subjective work graph as viewed by i, including a misreport attack by j to boost its score in BarterCast. Dotted edges indicate misreports.

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. The BASIC mechanism does not 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 via this attack. 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 the BASIC mechanism takes the maximum of two reports, agent i will believe j's report. As a result, agent k's score has decreased from 0 to -3, and agent j's score has increased from 0 to 3. Note that simply replacing the max-operator with the min-operator in the definition of the BASIC mechanism does *not* make it misreport-proof. Consider again Figure 4 (b). Using the min-operator prevents the attack where agent j exaggerates the amount of work he has done for agent k. However, now agent j can misreport the amount of work that k has done for j, for example it can report 0 instead of 2. Now, if the BASIC mechanism took the minimum instead of the maximum of two conflicting reports, the effect would be that after this particular attack, k score would decrease by 2 and j's score would increase by 2. Whether max or min is used, in both variants of the BASIC mechanism it is a dominant strategy to always report ∞ work performed, and 0 work consumed. Thus, a more sophisticated mechanism is necessary to defend against misreport attacks.

The DROP-EDGE mechanism ignores some of the information available to an agent, depending on context. Here, the "context" is the agent's current choice set C_i. If the agent ignores the reports from all agents currently inside the choice set, the resulting mechanism becomes misreport-proof.

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:*

$$\forall (j,k) | i \in \{j,k\} : w_i^D(j,k) = w_i^i(j,k) \tag{1}$$

$$\forall (j,k) | j,k \in C_i : w_i^D(j,k) = 0 \tag{2}$$

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

$$\forall (j,k) | k \in C_i, j \notin C_i : w_i^D(j,k) = w_i^j(j,k) \tag{4}$$

$$\forall (j,k) | j,k \notin C_i, i \notin \{j,k\} : w_i^D(j,k) = \max\{w_i^j(j,k), w_i^k(j,k)\} \tag{5}$$

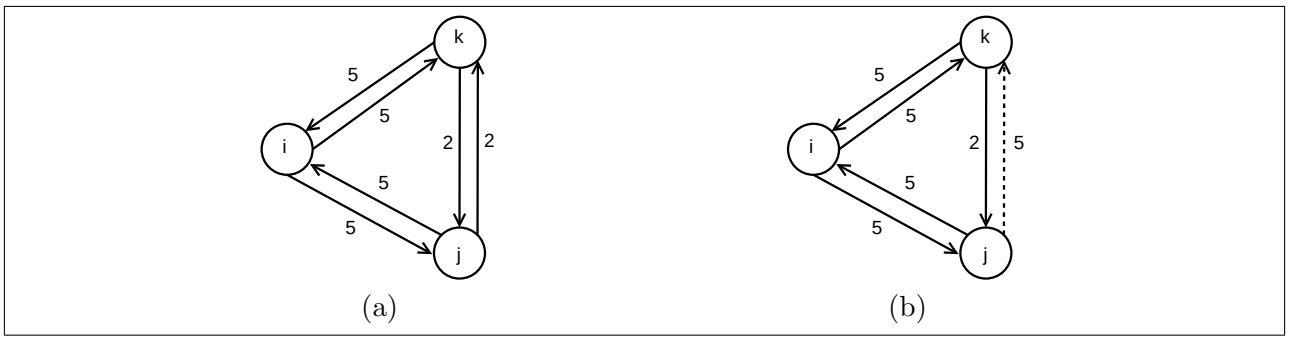


Figure 4: (a) A Work graph based on true reports. (b) The subjective work graph as viewed by i, including a misreport attack by j to decrease k's score and increase its own score when the Basic mechanism is used.

Missing reports in the max-operator are set to 0. *Agent j's score is* $S^D_{ij}(G_i, C_i) = MF_{G^D_i}(j,i) - MF_{G^D_i}(i,j)$.[5]

An agent takes its own information/experience over reports of others (1). Lines (2)-(4) implement the "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^D_i. An edge (j,k) is dropped completely if both j and k are inside C_i (4). In the case of two conflicting reports by two agents outside the choice set, the mechanism takes the maximum 5, thereby always using the most up-to-date information available. For an illustration of the Drop-Edge mechanism see Figure 5.

We make the following simple 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. □

Note that both the Basic mechanism and the Drop-Edge mechanism need to compute the maximum flow on a large work graph to determine the scores. In practice, however, running the max-flow algorithm on the full work graph may take too long due to the computational complexity of the max-flow algorithm. If we assume that the work graphs has at least as many edges as nodes, then all known algorithms have a running time that is at least quadratic in the number of nodes (see Goldberg et al. [13] for an overview of max-flow algorithms). The only exception is the Ford-Fulkerson algorithm which has a running time of $\mathcal{O}(Ef)$ when edge weights are integral, where E is the number of edges and f is the largest flow in the network. However, most work graphs in practice will have a relatively large number of edges because the applications that are suitable for accounting mechanisms generally lead to many short-term interactions between many agents. Thus, for graphs with thousands or millions of agents, even a running time of $\mathcal{O}(Ef)$ is prohibitive. On the other hand, if the distributed work graph is relatively dense, then most nodes are connected via short paths, and it may be sufficient to run a max-flow algorithm that only considers paths of a restricted length, and such algorithms run much faster in practice. Indeed, Piatek et al. [25] have found empirically, that 99% of the agents in a

[5] We do not need the max-flow algorithm to obtain misreport-proofness. However, we use it because it provides some additional protection against sybil and Byzantine attacks as we discuss in more detail in Section 4.3, and also makes the comparison with Basic easier.

P2P file sharing networks are connected via paths of length 1 or 2. For the experimental results we present in Section 5 we have used mechanisms with max-flow restricted to at most 1 hop (i.e., paths of length at most 2), and for the simulations we present in Section 6 we have used mechanism with max-flow restricted to at most 2 hops.

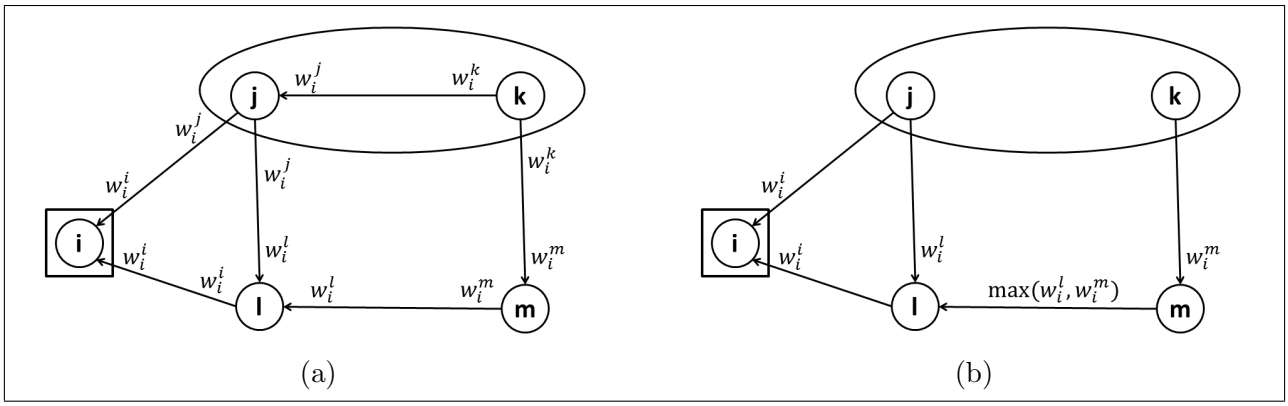


Figure 5: An illustration of the DROP-EDGE mechanism. We are showing subjective work graphs from agent i's perspective. The choice set is $C_i = \{j, k\}$. (a) Agent i's subjective work graph where each edge has two weights, one from each agent who knows about that edge. (b) Agent i's subjective work graph after the DROP-EDGE mechanism has been applied.

4 Theoretical Analysis

4.1 Information Loss of Drop-Edge

In the last section we have shown that by dropping some of the information based on context, the resulting mechanism becomes misreport-proof. However, this obviously comes at a cost because having more information about the past actions of other agents generally helps to better discriminate between cooperative and free-riding agents. We are interested in this trade-off between *informativeness* on the one side, and *misreport-proofness* on the other side. In this section we analyze the information loss of DROP-EDGE due to the discarded edges and show that it is small and vanishes in the limit as the number of agents in the network grows. Thus, the misreport-proofness of DROP-EDGE comes at a relatively small cost. Without misreport-proofness, strategic manipulations introduce an additional cost.

The following analysis is based on agent i's subjective work graph $G_i = (V_i, E_i, w_i)$. To isolate the question of "information loss" due to dropping some of the information, we only consider centralized protocols where all agents make reports to a centralized entity after ever time step, and we assume that agents do not perform any kind of manipulations. We let $G_i^D = (V_i, E_i, w_i^D)$ denote the modified graph after the DROP-EDGE mechanism has been applied to G_i. Note that which edges are dropped in G_i^D depends on which particular choice set C_i is chosen. Analogously, $G_i^B = (V_i, E_i, w_i^B)$ denotes the modified graph after the BASIC mechanism has been applied to G_i. Note that using a centralized information exchange protocol and assuming no manipulations, there won't be any conflicting reports in the subjective work graphs, and thus G_i^B represents the true (omniscient) work graph.

As a first step, we study the result of dropping edge on the net work information contained in the work graphs. Later, we add the use of max-flow to the analysis. For graph $G_i^D = (V_i, E_i, w_i^D)$, we

define the net work on edge (k,j) as $\tilde{w}_i^D(k,j) = w_i^D(k,j) - w_i^D(j,k)$ so that k's overall net work from i's perspective is $work_i(k, G_i^D) = \sum_{j \neq k} \tilde{w}_i^D(k,j)$. Analogously, for graph $G_i^B = (V_i, E_i, w_i^B)$, we let $\tilde{w}_i^B(k,j) = w_i^B(k,j) - w_i^B(j,k)$ and $work_i(k, G_i^B) = \sum_{j \neq k} \tilde{w}_i^B(k,j)$. Thus, the term $work_i(k, G_i^B)$ represents agent k's *true* net work, and $work_i(k, G_i^D)$ represents DROP-EDGE's approximation of agent k's net work, both according to i's subjective work graph.

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_i chosen uniformly at random with $|C_i| = m$ and $k \in C_i$:*

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

Proof.

$$\begin{aligned}
&\mathbb{E}_{C_i}[work_i(k, G_i^D)] && (6)\\
&= \mathbb{E}_{C_i}[\sum_{j \neq k} \tilde{w}_i^D(k,j)] && (7)\\
&= \sum_{j \neq k} \mathbb{E}_{C_i}[\tilde{w}_i^D(k,j)] && (8)\\
&= \sum_{j \neq k} \Big[\frac{(m-1)}{(n-1)} \cdot 0 + \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot \tilde{w}_i^B(k,j) \Big] && (9)\\
&= \big(1 - \frac{(m-1)}{(n-1)}\big) \cdot work_i(k, G_i^B)
\end{aligned}$$

For equation (9), consider edge (k,j). Because C_i is chosen uniformly at random with $k \in C_i$, the probability that j is also inside any random C is $\frac{m-1}{n-1}$. If k and j are inside C_i the edge gets dropped, otherwise $\tilde{w}_i^B(k,j)$ 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.

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_i chosen uniformly at random with $|C_i| = m$, it holds that:*

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

We now turn our attention to the approximation ratio of the scores computed by Drop-Edge when the max-flow algorithm is used. The first theorem is with regard to running the full max-flow algorithm, however, for the analysis, we need to consider max-flows restricted to a certain number of hops. We let $MF_G^h(i,j)$ denote the max-flow from node i to j in graph G with exactly h hops. We let $S_{ij}^{D,h}(G_i, C_i)$ denote the score computed by Drop-Edge for h hops, i.e., $S_{ij}^{D,h}(G_i, C_i) = MF_{G_i^D}^h(j,i) - MF_{G_i^D}^h(i,j)$. Analogously, $S_{ij}^{B,h}(G_i, C_i)$ is the score computed by the BASIC mechanism using max-flow with exactly h hops.

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

$$E_{C_i}[S^D_{ik}(G_i, C_i)] = S^{B,0}_{ik}(G_i, C_i) + \sum_{h=1}^{n-m-1} \prod_{p=1}^{h} \left(\frac{n-m-p}{n-1-p}\right) \cdot S^{B,h}_{ik}(G_i, C_i).$$

Proof.

$$E_{C_i}[S^D_{ik}(G_i, C)] = E_{C_i}[MF_{G_i^D}(k,i) - MF_{G_i^D}(i,k)] \tag{10}$$

$$= \sum_{h=0}^{n-m-1} E_{C_i}[MF^h_{G_i^D}(k,i) - MF^h_{G_i^D}(i,k)] \tag{11}$$

$$= E_{C_i}[MF^0_{G_i^D}(k,i) - MF^0_{G_i^D}(i,k)] \tag{12}$$

$$+ \sum_{h=1}^{n-m-1} E[MF^h_{G_i^D}(k,i) - MF^h_{G_i^D}(i,k)] \tag{13}$$

$$= S^{B,0}_{ik}(G_i, C_i) \tag{14}$$

$$+ E_{C_i}[MF^1_{G_i^D}(k,i) - MF^1_{G_i^D}(i,k)] \tag{15}$$

$$+ \sum_{h=2}^{n-m-1} E_{C_i}[MF^h_{G_i^D}(k,i) - MF^h_{G_i^D}(i,k)] \tag{16}$$

$$= S^{B,0}_{ik}(G_i, C_i) + \left(\frac{n-m-1}{n-2}\right) \cdot S^{B,1}_{ik}(G_i, C_i) \tag{17}$$

$$+ \left(\frac{n-m-1}{n-2}\right) \cdot \left(\frac{n-m-2}{n-3}\right) \cdot S^{B,2}_{ik}(G_i, C_i) \tag{18}$$

$$+ \sum_{h=3}^{n-m-1} E_{C_i}[MF^h_{G_i^D}(k,i) - MF^h_{G_i^D}(i,k)] \tag{19}$$

$$= S^{B,0}_{ik}(G_i, C_i) + \sum_{h=1}^{n-m-1} \prod_{p=1}^{h} \left(\frac{n-m-p}{n-1-p}\right) \cdot S^{B,h}_{ik}(G_i, C_i) \tag{20}$$

In Equation (12) we isolated the expectation of the 0-hop max-flow terms which consider the direct paths between i and k and thus do not involve dropped edges, and consequently Equation (14) follows because $S^{D,0}_{ik} = S^{B,0}_{ik}$. In Equation (15) we isolated the expectation of the 1-hop max-flow terms, i.e., the flows along all paths of length 2 between i and k. Because C_i was chosen uniformly at random, for any of the 1-hop paths between i and k the probability that the intermediate node lies outside of C_i is $\frac{n-m-1}{n-2}$ and Equation (17) follows. The final expression follows from analogous reasoning for all h-hop max-flows. □

Remember that running the full max-flow algorithm may be computationally prohibitive, which is why we use max-flow algorithms restricted to at most 1 or 2 hops in our experiments. We let $S^{D,\leq 1}_{ik}$ denote the scores obtained by the Drop-Edge mechanism when max-flow is restricted to at most 1 hop, and $S^{B,\leq 1}_{ik}$ analogously for the Basic mechanism. The following corollary tells us the accounting accuracy of Drop-Edge using a max-flow algorithm restricted to at most 1 hop:

Corollary 2. *If the max-flow algorithm is restricted to a most 1 hop, 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 chosen uniformly at random*

with $|C_i| = m$ *and* $k \in C_i$*:*

$$\frac{E_{C_i}[S_{ik}^{D,\leq 1}(G_i, C_i)]}{S_{ik}^{B,\leq 1}(G_i, C_i)} \geq (\frac{n-m-1}{n-2}).$$

The theoretical results in this section bound the accuracy in expectation over choice sets and do not directly pertain to accuracy with respect to selecting the right agent from a given choice set. Moreover, the approximation ratios for the max-flow based mechanism do not directly compare the scores obtained when using max-flow to the scores obtained if the net work were considered directly. In Sections 5 and 6 we see that the information-loss of DROP-EDGE is indeed small in practice, even for small graphs, and that the mechanism leads to very good efficiency.

4.2 Sybil-Proofness

Now we turn our attention to a class of attacks called *sybil attacks*, where an agent introduces sybil nodes (fake agents) into the network to manipulate the accounting mechanism.

4.2.1 Preliminaries

We distinguish between *passive* sybil attacks, where the sybils themselves may consume but not perform work, and *active* sybil attacks, where the sybils themselves also perform work.[6]

Definition 11. ***(Passive Sybil Attack)*** *A passive sybil attack by agent* j *is a tuple* $\sigma_j = (V_s, E_s, w_s)$ *where* $V_s = \{s_{j_1}, s_{j_2}, ...\}$ *is a set of sybils,* $E_s = \{(x, y) : x, y \in S \cup \{j\}\}$*, and* w_s *are the edge weights for the edges in* E_s*. Applying the sybil attack* σ_j *to agent* i*'s subjective work graph* $G_i = (V_i, E_i, w_i)$ *results in a modified work graph* $G_i \downarrow \sigma_j = G_i' = (V_i \cup V_s, E_i \cup E_s, w')$ *where* $w'(e) = w_i(e)$ *for* $e \in E_i$ *and* $w'(e) = w_s(e)$ *for* $e \in E_s$*.*

Definition 12. ***(Active Sybil Attack)*** *An active sybil attack by agent* j *is a tuple* $\sigma_j = (V_s, E_s, E_s^{\rightarrow}, w_s, w_s^{\rightarrow})$ *where* $V_s = \{s_{j_1}, s_{j_2}, ...\}$ *is a set of sybils,* $E_s = \{(x, y) : x, y \in V_s \cup \{j\}\}$*,* $E_s^{\rightarrow} = \{(x, y) : x \in V_s, y \notin V_s \cup \{j\}\}$ *are edges indicating work done by the sybils,* w_s *are the edge weights for the edges in* E_s*, and* $w_s^{\rightarrow}$ *are the edge weights for edges in* $E_s^{\rightarrow}$*. Applying the sybil attack* σ_j *to agent* i*'s subjective work graph* $G_i = (V_i, E_i, w_i)$ *results in a modified work graph* $G_i \downarrow \sigma_j = G_i' = (V_i \cup V_s, E_i \cup E_s \cup E_s^{\rightarrow}, w')$ *where* $w'(e) = w_i(e)$ *for* $e \in E_i$ *and* $w'(e) = w_s(e)$ *for* $e \in E_s$ *and* $w'(e) = w_s^{\rightarrow}(e)$ *for* $e \in E_s^{\rightarrow}$*.*

So far we have only defined the strategy space for passive and active sybil attacks. Such an attack can only be "beneficial" if the attacking agent is better of after the attack than before. Remember that S_0^M denotes the default score that accounting mechanism M assigns to an agent about which no information is available.

Definition 13. ***(Beneficial Sybil Attack)*** *Given accounting mechanism* M *and work graph* $G_i = (V_i, E_i, w_i)$*, a beneficial (passive or active) sybil attack* σ_j *by agent* $j \in V_i$ *such that* $G_i' = \sigma_j(G_i)$ *is one where option (1), (2), or (3) holds:*

(1) $\exists C_i$ *s.t.* $j \in C_i$ *and* $S_{ij}^M(G_i, C_i) < S_{ij}^M(G_i', C_i)$

(2) $\exists k \in V_i \backslash \{j\}$ *and* $\exists C_i$ *with* $j, k \in C_i$*:* $(S_{ij}^M(G_i, C_i) < S_{ik}^M(G_i, C_i)) \quad \wedge \quad (S_{ij}^M(G_i', C_i) > S_{ik}^M(G_i', C_i))$

[6] Note that our definitions differ from previous definitions of sybil attacks on reputation mechanisms (see [5] and [6]), in particular the one for active sybil attacks where we also allow sybil agents to perform work.

(3) $\exists s \in V_s : (\exists C_i \text{ with } s \in C_i : S^M_{is}(G'_i, C_i) > S^M_0) \quad \wedge \quad (\forall C_i \text{ with } j \in C_i : S^M_{ij}(G'_i, C_i) \geq S^M_{ij}(G_i, C_i))$

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 score strictly higher than S^M_0 without decreasing its own score.

To study the sybil attacks as introduced so far, we do not need a dynamic, multi-step analysis. Given work graph G_i, an attacking agent can do multiple things, e.g., add sybils to the network, make multiple false reports about these sybils, etc. We model all of this as happening in one step, inducing a new subjective work graph G'_i. However, when we consider long-term effects of an attack, we also have to look at what happens when certain attacks are repeated over and over again. How beneficial a sybil attack really is, depends on the trade-off between the amount of work necessary to perform the attack, and the amount of "free" work the agent can consume as a result of the attack. We will distinguish between *long-term* beneficial sybil attacks on the one side, where the ratio between work performed and consumed goes towards infinity as the attack is repeated, and *short-term* beneficial sybil attacks on the other side, where that ratio is bounded by a constant.

Definition 14. ***(Long-term vs. Short-term Beneficial Sybil Attacks)*** *Given accounting mechanism M and work graph $G_i = (V_i, E_i, w_i)$, assume agent $j \in V_i$ performs a (passive or active) sybil attack σ_j such that $G'_i = \sigma_j(G_i)$. Let σ_j^n denote an $n-$times-repetition of the sybil attack. Let $\omega^-(\sigma_j^n)$ denote the amount of work involved in performing σ_j^n, and let $\omega^+(\sigma_j^n)$ denote the amount of work that agent j or any of its sybils will be able to consume. We call σ_j a:*

- ***long-term beneficial sybil attack:*** *if $\omega^+(\sigma_j^n) > 0$ and $\omega^-(\sigma_j^n) = 0$ or $\lim_{n\to\infty} \frac{\omega^+(\sigma_j^n)}{\omega^-(\sigma_j^n)} = \infty$.*
- ***short-term beneficial sybil attack:*** *if $\omega^+(\sigma_j^n) > 0$ and $\omega^-(\sigma_j^n) > 0$ and $\exists c \in \mathbb{R}_{\geq 0}$: $\lim_{n\to\infty} \frac{\omega^+(\sigma_j^n)}{\omega^-(\sigma_j^n)} \leq c$.*

A long-term beneficial (passive) sybil attack on the Basic and Drop-Edge mechanisms is illustrated in Figure 6 where condition (3) of Definition 13 holds. In this example, agent j has already performed/consumed 10 units of work for/from agent i (such that agent i believes agent j's reports about other agents). To perform the sybil attack, agent j creates a set of sybils and falsely reports to i that these sybils have performed 10 units of work for j, such that i now assigns a score strictly higher than S^M_0 to the sybil nodes, in particular, both the Basic and Drop-Edge mechanisms assign a score of 10 to the sybil nodes. Each sybil agent can now exploit its score and consume some work from i (assuming, at some point, the sybils will be in i's choice set with other agents that have a lower score). Once the sybils' scores are "used up", j can simply create another sybil s'_j and repeat the attack ad infinitum. This attack is powerful because it only requires a *passive* sybil attack that involves no work to be performed by j or any of its sybils. In general, however, passive attacks may be long-term or short-term beneficial, and active attacks may also be long-term or short-term beneficial.

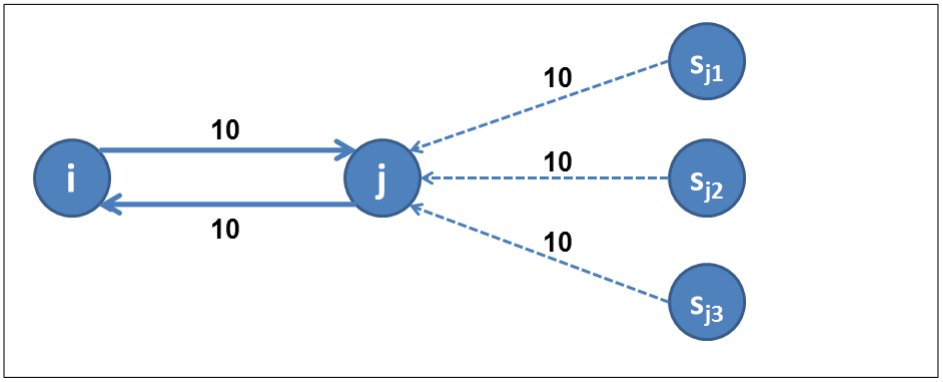


Figure 6: A sybil attack where agent j generates many sybils, then does a little bit of work for i and then provides its sybils with positive scores.

Definition 15. ***(Sybil-Proofness)*** *An accounting mechanism M is*

- *sybil-proof against long-term beneficial sybil attacks, if for every work graph G_i there exists no (passive or active) long-term beneficial sybil attack on M with respect to G_i.*
- *sybil-proof against short-term beneficial sybil attacks, if for every work graph G_i there exists no (passive or active) short-term beneficial sybil attack on M with respect to G_i.*

Before we can formally prove our first impossibility theorem regarding sybil-proofness, we introduce a series of natural assumptions regarding accounting mechanisms. First, we assume 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 change the scores of other agents. More formally:

Definition 16. ***(Independence of Disconnected Agents)*** *An accounting mechanism M satisfies independence of disconnected agents, if for 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, where $G'_i = (V'_i, E'_i, w'_i)$ denotes 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 $\forall j \in V'_i : S^M_{ij}(G_i, C_i) = S^M_{ij}(G'_i, C'_i)$ for all $G''_i = (V''_i, E''_i)$ the following holds:*

Furthermore, we will assume that a priori, the accounting mechanism does not put more or less trust into any agent in the network. More formally, we only consider mechanisms that, for any renaming of the agents in the network, return the same scores, i.e., they are *symmetric*[7]:

Definition 17. ***(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 $f(i) = i$:*

$$\forall j \in V_i \setminus \{i\} : S^M_{ij}(G_i, C_i) = S^M_{if(j)}(G'_i, C'_i).$$

[7]Note that in the context of reputation mechanisms, *symmetry* typically corresponds to globally consistent, or *objective* reputation values, where every agent in a network has the same view on each other agent's reputation, in contrast to *asymmetric* mechanisms that allow for *subjective* reputation values. Cheng et al. [5] have shown that no symmetric and sybil-proof reputation mechanisms exists. According to Cheng et al.'s definition, our accounting mechanisms would all be called "asymmetric" because they all inherently lead to subjective scores because the scores are computed based on subjective work graphs. However, what we mean by "symmetry" is something different, namely that from each individual agent's perspective, the rest of the network is symmetric.

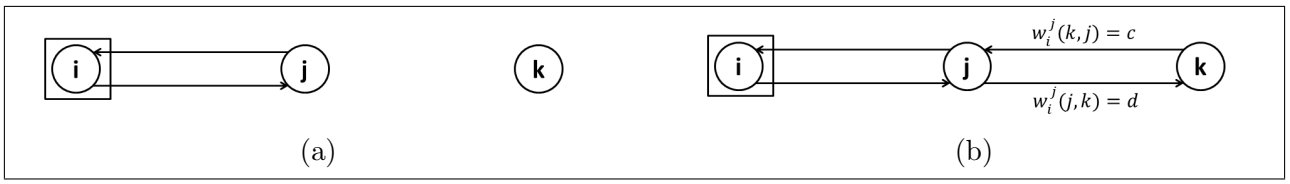


Figure 7: An illustration of the single-report-responsiveness property: there exists a subjective work graph G_i, e.g., the one show in (a), such that a single positive report by j about k, as shown in (b), leads to $S_{ik}^M(G_i, C_i) > S_0^M$, and a single negative report by j about k leads to $S_{ik}^M(G_i, C_i) < S_0^M$.

For the design of sybil-proof accounting mechanisms, we want to exclude any "trivial" accounting mechanisms that assign the same or random scores to every agent, as well as mechanisms that ignore all information except for their own direct experiences. Assuming *single-report responsiveness* excludes these mechanisms:

Definition 18. ***(Single-Report Responsiveness Property)*** *Let $dist(i,j)$ denote the length of the shortest path between i and j. 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 no path is 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:*

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

and analogously, a constant $d \in \mathbb{R}_{>0}$ with $w_i'^j(j,k) = d$, such that:

$$S_k^M(G_i', C_i') < S_0^M.$$

An illustration of the single-report responsiveness property is depicted in Figure 7. What this property says is that there exists a situation (i.e., a specific work graph), where i has no information about k, and a *single* positive report by agent j about agent k can increase the score that agent i assigns to agent k above S_0^M, and that a *single* negative report by agent j about agent k can decrease the score that agent i assigns to agent k below S_0^M. Thus, if an agent was previously unknown to me, a single positive or negative report about that agent can potentially change my evaluation of that agent.

4.2.2 Impossibility of Sybil-Proofness

Note that both the Basic and the Drop-Edge mechanism satisfy the independence of disconnected agents, are symmetric, and satisfy the single-report responsiveness property. We have already shown (Figure 6) that both mechanisms are susceptible to sybil attacks. We will now show that this is generally unavoidable:

Theorem 3. *For every accounting mechanism M that satisfies independence of disconnected agents, is symmetric, has the single-report responsiveness property, and is misreport-proof, there exists a (passive) long-term beneficial sybil attack.*

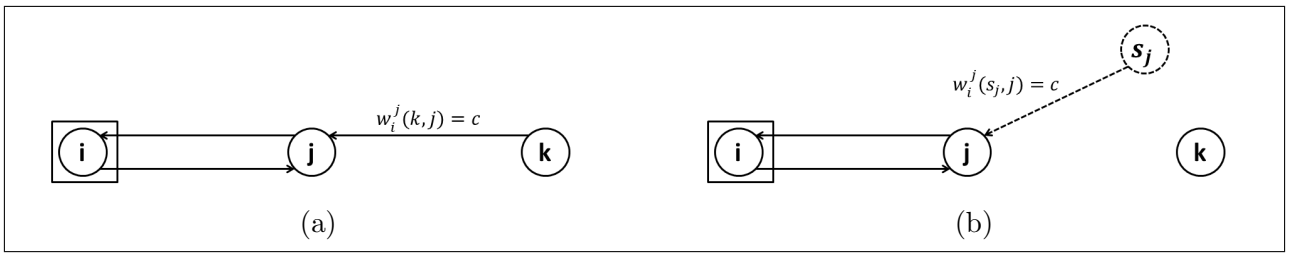


Figure 8: An illustration of the long-term beneficial sybil attack used in the proof for Theorem 3. If the mechanism is misreport-proof, then j has no disadvantage from making a truthful report $w_i^j(k,j) = c$ about another agent k. If the mechanism is also symmetric and satisfies independence of disconnected agents, then j can create sybil s_j, and make a report $w_i^j(s_j,j) = c$ about s_j, also without a disadvantage to j. Thus, if originally, the positive report about k lead to a positive score for k, then now the sybil node s_j has a positive score as well, and the attack does not require any actual work to be performed by j.

Proof. Let's assume accounting mechanism M satisfies the single-report responsiveness property. Thus, there exists a graph G_i and nodes i, j and k as described in Definition 18, for example like the one depicted in Figure 8 (a). 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 of disconnected agents, 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. Now, assume that agent k performs c units of work for j (as needed for the single-report responsiveness property) and agent j makes a truthful report to i about this interaction, leading to subjective work graph G_i'' such that $S_{ik}^M(G_i'', C_i) > S_0^M$. Here it is essential that M is a misreport-proof mechanism. Because M is misreport-proof, we know that agent j has no disadvantage from reporting truthfully, i.e., for any possible misreport that would lead to subjective work graph G_i''' it holds that: $\forall C_i$ with $j \in C_i : S_{ij}^M(G_i'', C_i) \geq S_{ij}^M(G_i''', C_i)$. Because M is symmetric, we can apply a graph isomorphism f to G_i'' that only switches the labeling of s_j and k but nothing else. Thus, there exists a report that j can make about s_j with $w_i'^j(s_j, j) = c$ leading to graph G_i^* such that $S_{is_j}^M(G_i^*, C_i) > S_0^M$ (see Figure 8 (b)). As before with node k, because of misreport-proofness, we know that agent j has no disadvantage from making this report. Thus, property (3) of Definition 13 is satisfied and because the attack itself involves no work, this constitutes a long-term beneficial sybil attack on M. □

4.2.3 (Im-)Possibility of K-Sybil-Proofness

In this section we explore whether we can achieve any kind of formal sybil-proofness guarantees, despite the strong negative results from the last section. 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 or negative reports about an agent, before the mechanism assigns a score distinct from S_0^M to that agent. This leads to the following generalization of the responsiveness property:

Definition 19. ***(K-Report Responsiveness Property)*** *Let $dist(i,j)$ denote the distance between two nodes i and j in a graph. An accounting mechanism M has the K-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 node l and a set of nodes V_K with $|V_K| = K$, such that $\forall k \in V_K : dist(i,k) = dist(k,i) = 1$ and $dist(i,l) =$*

$dist(l,i) = \infty$ (i.e., nodes i and all nodes in V_K are neighbors in G_i and no path is connecting i and l), and there exists a graph $G'_i = (V'_i, E'_i, w'_i)$ with $V'_i = V_i$, $E'_i = E_i \cup \{(k,j),(j,k) | k \in V_K\}$, and $w'_i(e) = w_i(e)$ for all $e \in E_i \setminus \{(k,j),(j,k) | k \in V_K\}$, and there exists a constant $c \in \mathbb{R}_{>0}$ with $w'^{ij}_i(k,j) = c$ for all $k \in V_K$, such that:

$$S^M_{ik}(G'_i, C'_i) > S^M_0$$

and a constant $d \in \mathbb{R}_{>0}$ with $w'^{ij}_i(j,k) = d$ for all $k \in V_K$, such that:

$$S^M_{ik}(G'_i, C'_i) < S^M_0.$$

Obviously, performing a sybil attack against a mechanism that does not have the single-report responsiveness property, but the K-report responsiveness property is more difficult, and the attack would require additional work, either by the sybil agents or by the manipulating agent itself. We can now define a corresponding, weaker notion of sybil-proofness:

Definition 20. ***(K-Sybil-Proofness)*** *An accounting mechanism M is* K-Sybil-proof against long-term beneficial sybil attacks *if, for every work graph G_i, there does not exist a long-term beneficial sybil attack with K or fewer sybils for M; it is* K-Sybil-proof against short-term beneficial sybil attacks *if there does not exist a short-term beneficial sybil attack with K or fewer sybils for M.*

Theorem 4. *No accounting mechanism that is K-report responsive, is symmetric, and satisfies independence of disconnected agents is K-sybil-proof against short-term beneficial sybil attacks.*

Proof. Let's assume accounting mechanism M is symmetric, satisfies independence of disconnected agents, and is K-report responsiveness. Then there exists a subjective work graph G_i and nodes l and V_K as described in Definition 19. In particular, if all agents k in V_K make a report about the edge (k,l) with weight c to agent i, then the resulting score for agent l is greater than S^M_0, i.e.,: $S^M_{il}(G'_i, C'_i) > S^M_0$. Now, let's remove one agent k^* from the set V_K, leading to subjective work graph G''_i. Now, $S^M_{il}(G''_i, C''_i) = S^M_0$ again. Now we let agent j create a sybil agent s_j. Because of the independence of disconnected agents, this does not change any of the scores. Now assume that agent s_j performs the same number of units of work for i that previously k^* had performed. Now, from i's perspective agents k^* and s_j look the same. Thus, because M is symmetric, if agent s_j now makes a report about edge (l, s_j) with weight c, then $S^M_{il}(G'''_i, C'''_i) > S^M_0$. Without loss of generality, we can assume that $l = j$. Thus, by creating just one sybil s_j agent j has performed a short-term beneficial sybil attack. □

We will now show how to turn any accounting mechanism into a K-report responsive mechanism that is K-sybil-proof against long-term beneficial sybil attacks:

Definition 21. ***(K-Elimination-Wrapper)*** *A K-Elimination-Wrapper W takes as input an accounting mechanism M, a subjective work graph $G_i = (V_i, E_i, w_i)$, and a choice set C_i, and determines the scores $S^W_{ij}(M, G_i, C_i)$ for each agent $j \in C_i$, as viewed by agent i. Let $\mathcal{P}(V_i)$ denote the powerset of V_i, and let $\mathcal{P}_{\leq K}(V_i)$ denote the set of subsets of $\mathcal{P}(V_i)$ of cardinality less than or equal to K. We let $G_i \setminus X$ denote the subjective work graph that results from taking G_i and removing all nodes in X from V_i. As before, we let $S^M_{ij}(G_i, C_i)$ denote the scores according to the accounting mechanism M. The wrapper scores $S^W_{ij}(M, G_i, C_i)$ are computed as follows:*

$$S^W_{ij}(M, G_i, C_i) = \min_{X \in \mathcal{P}_{\leq K}(V_i)} \{S^M_{ij}(G_i \setminus X, C_i \setminus X)\}.$$

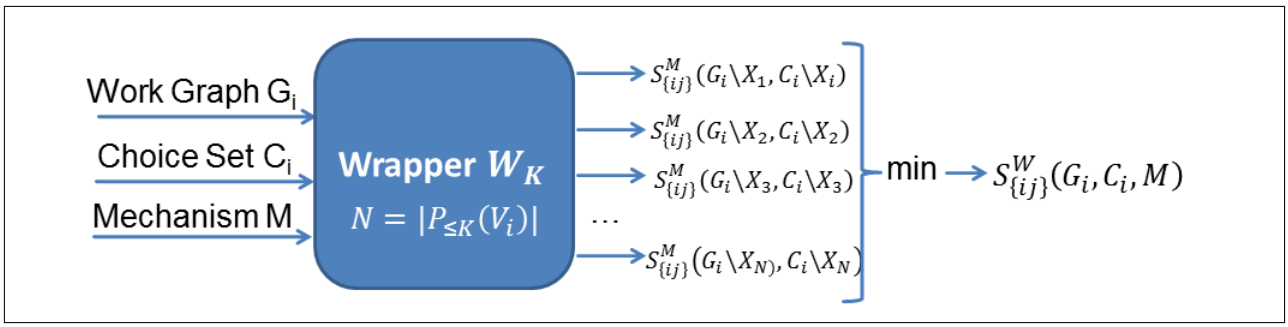


Figure 9: The K-Elimination-Wrapper.

Theorem 5. *A K-elimination-wrapper applied to any accounting mechanism leads to an accounting mechanism that is K-sybil-proof against long-term beneficial sybil attacks.*

Proof. Let's assume this is not true, i.e., there exists a sybil attack by some agent j that involve less than or equal to K sybils and is long-term beneficial. Note that the K-elimination wrapper iteratively removes all subsets of agents of size K or less from G_i, computes all scores without those subsets, and ultimately takes the minimum. Thus, in one of those iterations, all of j's K sybils will be removed from G_i, and the resulting score will be part of the overall minimization of the wrapper. Thus, if agent j's score before the sybil attack was lower than before the sybil attack, then the wrapper will take the score from before the attack, rendering the sybil attack useless. This excludes options (1) and (3) from the set of beneficial sybil attacks (see Definition 13) where the goal of the sybil attack was to increase agent j's or one of the sybils' scores. This only leaves option (2), which requires an active sybil attack, where the sybil agents themselves perform work and make misreports, such that after the sybil attack, an agent k, i.e., one of the other agents in the network, now has a lower score than before, such that the relative ordering of j and k has changed. If this attack is indeed successful (i.e., j and k are inside the same choice set and now j gets allocated instead of k), then j gets to consume some units of work "for free". However, note that after consuming a certain amount of work, j's score is lowered again, and at some point, j's score will be lower than k's score again. Thus, now another sybil attack would be necessary, which again would require the sybil agents to perform work. Thus, the amount of free work resulting from this sybil attack is bounded: for every x units of "free" work, the sybil attack requires a certain fixed amount of work as well. Thus, the sybil attack can at best be short-term beneficial, but not long-term beneficial. □

Note that using the K-elimination-wrapper does not provide any robustness against short-term beneficial sybil attacks, and even achieving K-sybil-proofness against long-term beneficial sybil attacks comes at a cost: the resulting mechanism is only K-report-responsive and ignores a larger part of the available information compared to a single-report responsive mechanism. Assuming random interactions between peers, the probability of having K reports about an agent decreases exponentially in K. Thus, real-world system designers face an important trade-off between (limited) robustness against sybil attacks on the one side, and informativeness on the other side. The decision regarding this trade-off can depend on many factors. For example, in some domains, creating one or two sybils may be relatively cheap, but creating more sybils could become very expensive (e.g.,. obtaining multiple IP addresses). If this is in fact the case, then a 3-sybil-proof mechanism might provide good robustness in that particular domain. Furthermore, in some domains the interactions between peers are not random, but highly clustered (e.g., in P2P file sharing communities with similar taste

preferences). Thus, in these domains it might be reasonable to assume that each agent has an average of K reports about each other agent, and thus, even after applying a K-elimination-wrapper, the resulting scores will still be informative enough. In future work, we will analyze this trade-off in more detail (analytically and experimentally).

4.3 The Role of the Max-Flow Algorithm

We have shown that we cannot achieve fully sybil-proof accounting mechanisms, and even limited robustness comes at a high price. One interesting way to address this problem in practice is the application of the max-flow algorithm inside an accounting mechanism. In fact, both BarterCast [22] and Drop-Edge [31] use max-flow. 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 without providing any additional formal guarantees. However, using max-flow provides additional robustness against sybil attacks in practice: max-flow bounds the influence of any agent by the total amount of work performed by that agent itself. This limits the power of sybil attacks, making them more costly and thus less attractive for the attacking agent.[8] Furthermore, max-flow is also useful to protect against Byzantine agents, i.e., agents that try to harm the network or specific agents in the network. For example, if 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. Using max-flow makes Byzantine attacks much more difficult and costly for the attacking agent, thereby effectively preventing them in practice.

5 Experimental Analysis: Discrete Simulations

In this section, we compare the mechanisms empirically via a discrete, round-based simulation to understand the trade-offs that are made in the Basic and Drop-Edge mechanisms in practice. Remember that we consider both centralized and decentralized information exchange protocols. For the decentralized version, we simulate the BarterCast information exchange protocol. Consequently, when referring to the centralized versions of the mechanisms, we write Basic and Drop-Edge as before, and when referring to the decentralized versions we write BarterCast-Basic and BarterCast-Drop-Edge. Considering the centralized version of each mechanism helps isolate the effect of the message exchange protocol from the mechanism itself.

5.1 Experimental Set-up

We simulate a generic distributed work system with 100 agents and discrete time steps, i.e.., this is a simulation without any particular application in mind. 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

[8] 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 (cf. Friedman et al. [11] or Tang et al. [33]), 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.

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.

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 aspect of the simulation is motivated by similar allocation rules used in P2P file sharing (e.g., optimistic unchoking in BitTorrent). For the decentralized information exchange protocol, every agent contacts one other agent at random in each round, to exchange messages about direct experiences in the network. Agents exchange reports about the last 5 agents they have interacted with and the 5 agents that have uploaded the most to them. For the centralized version, we assume the existence of a center, collecting reports (which may still be untruthful) and making them immediately available. For Basic and BarterCast-Basic, the strategic agents perform the optimal misreport manipulation, i.e., always reporting they have consumed 0 units of work and contributed ∞ units of work. Unless otherwise noted, we run each simulation for 100 time steps and record the work contributions and consumptions (averaged over 10 trials).

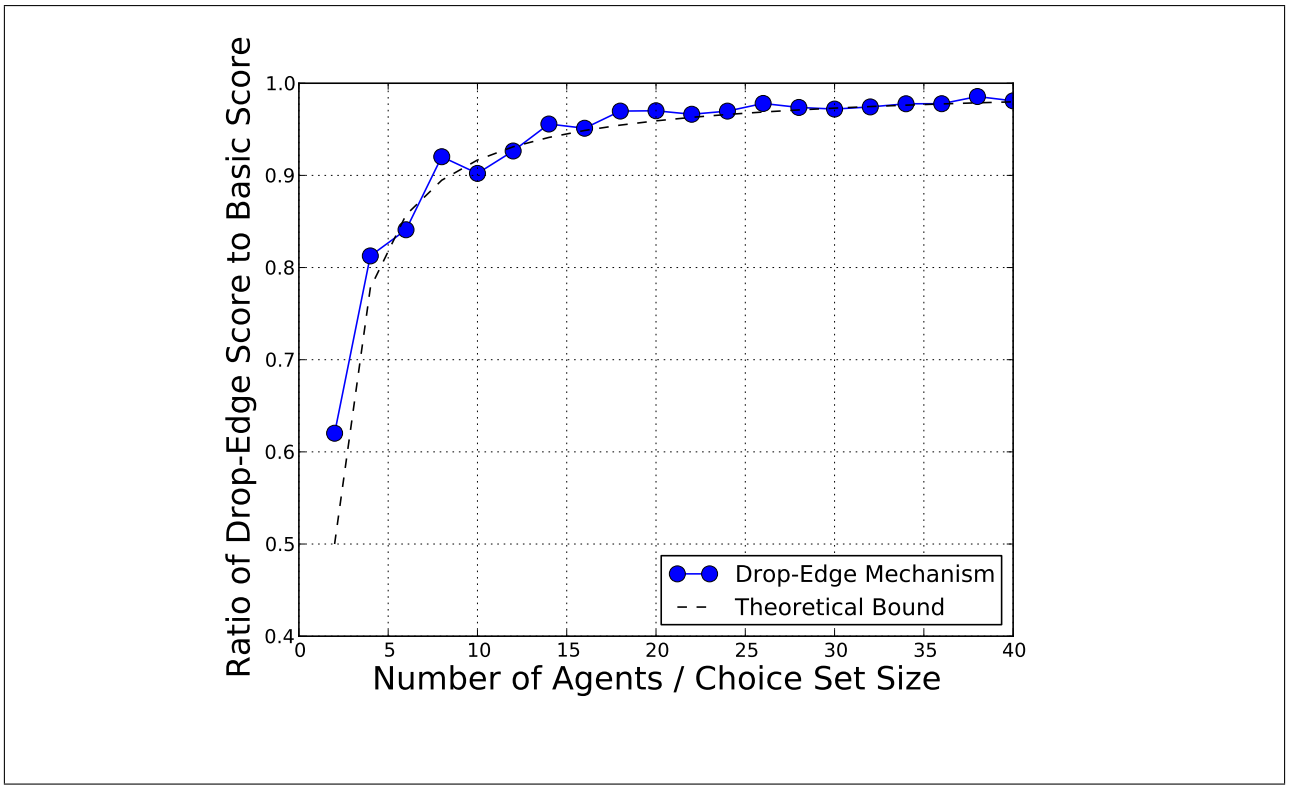


Figure 10: The ratio of the Drop-Edge scores and the Basic scores depending on network size. As the number of agents in the network grows relative to the choice set size, the approximation ratio approaches 1 and the information loss of Drop-Edge vanishes.

5.2 Information Loss of Drop-Edge

We first verify our theoretical results on information loss, in particular, that the information loss of Drop-Edge vanishes as the number of agents in the network gets large relative to the size of the

choice sets. To isolate the effect of information loss due to dropped edges, we simulate a network without strategic agents, and compare the scores obtained by the centralized DROP-EDGE mechanism with those obtained by the centralized BASIC mechanism (that does not drop edges). Fixing a choice set size of $m = 5$ and free-rider agent fraction $\beta = 0.5$, 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 BASIC mechanism. Averaging over all agents and choice sets, we find that our empirical results closely match the theoretical results (Corollary 2). In Figure 10 we plot the ratios of the DROP-EDGE scores and the BASIC scores (or the "true" scores) for different network sizes. We see that the approximation ratio approaches 1 as the number of agents in the network grows relative to the choice set size.

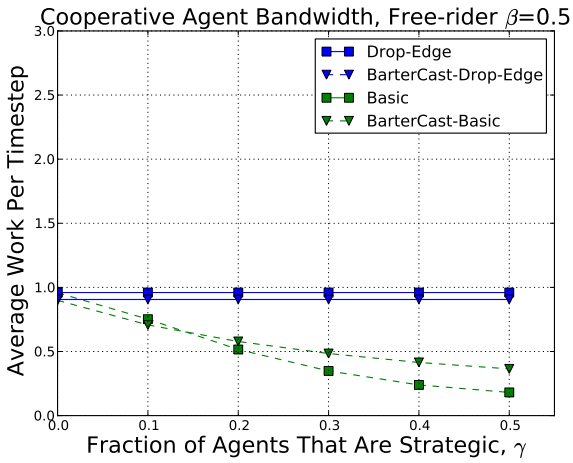
5.3 Efficiency Results

We now consider direct measures of performance. First, we measure the mechanisms' performance without strategic agents, to isolate their effectiveness as algorithms in aggregating information and promoting good decisions. Consider the graphs in Figure 11 (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 BASIC is using, and no strategic agents are present that could harm the BASIC mechanism. We see that the efficiency is indeed higher under both versions of the BASIC mechanism, but only minimally so (less than 5% difference).

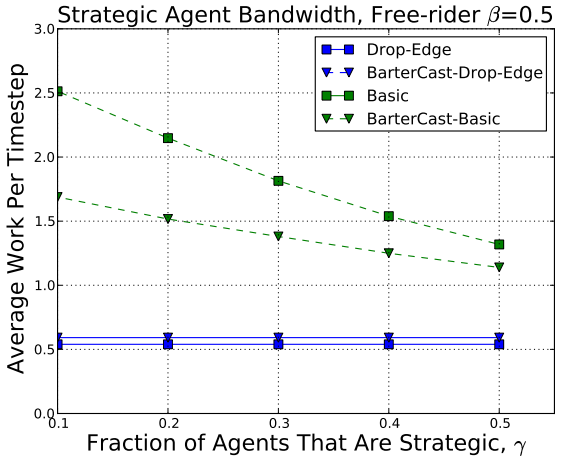
The more interesting analysis concerns the overall efficiency with strategic agents present. The *efficiency* of a particular agent type is defined to be the average amount of work received by that type of agent per time step. *It is our goal to maximize the efficiency of the cooperative agents and to 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 11 (a),(b) and (c), to analyze the relative efficiency of all agent types under the two 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 11(b), we clearly see that strategic agents are able to sharply increase their performance compared to the other agents (see Figures 11(a) and (c)) by misreporting under the BASIC mechanism. This effect is particularly high when only a few strategic agents are in the system. With 10% strategic agents, the performance of a strategic agent is 3 times as high as that of the other agents under the decentralized BARTERCAST-BASIC mechanism, and more than 5 times as high under the centralized BASIC mechanism. With the BASIC mechanism, 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.

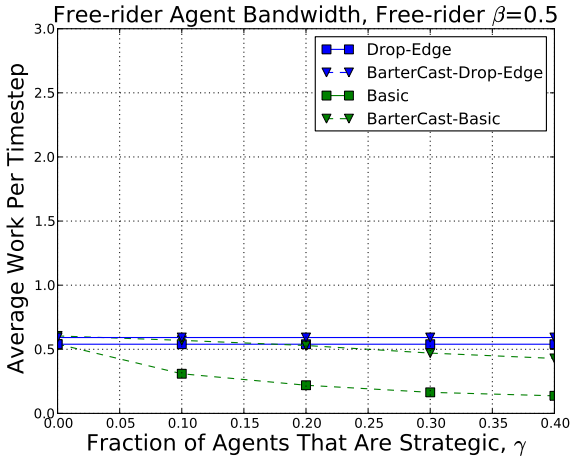
In practice, strategic misreports may also occur under DROP-EDGE even though such behavior is not useful for an agent. We have tested DROP-EDGE in settings with strategic agents (not plotted) and although the efficiency of the cooperative agents decreases slightly as the proportion of strategic agents increases, DROP-EDGE continues to clearly outperform the BASIC mechanism. 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 12 (a), we see that the benefit that strategic agents gain from misreporting in the BASIC mechanism gets even larger over time. Compare this against Figure 12 (b), which presents results for DROP-EDGE. 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, which is the ultimate goal, because they also perform



(a)

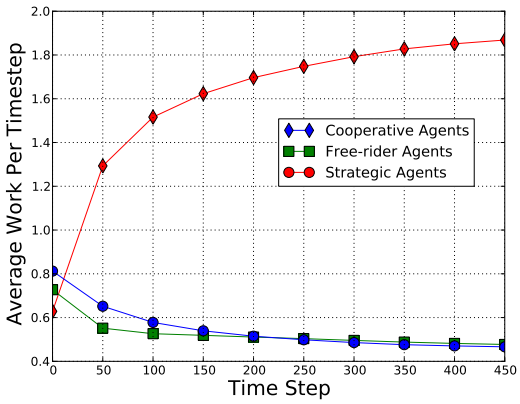


(b)

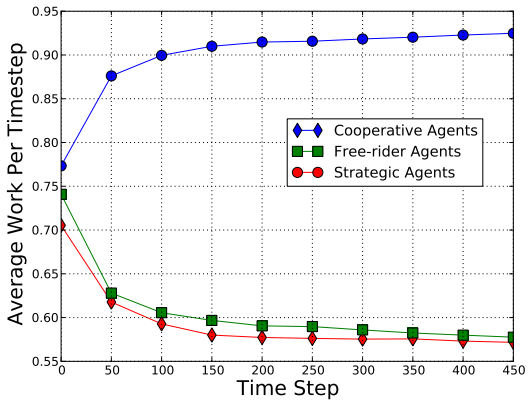


(c)

Figure 11: Average work consumption per time step, using the BASIC and the DROP-EDGE mechanisms. The fraction of free-riding agents is $\beta = 0.5$.



(a) The BASIC Mechanism



(b) The DROP-EDGE Mechanism

Figure 12: Evolution of average work consumption over time.

exactly twice as much work.

To summarize, in this section we have shown that the good approximation of the scores in DROP-EDGE also translates into good system efficiency. When strategic agents are present, the *Drop-Edge* mechanism clearly outperforms the BASIC mechanism: cooperative agents have higher efficiency, while free-riding agents have lower efficiency. We have shown that the magnitude of this effect even grows over time. Thus, we believe that using DROP-EDGE over BASIC in a real system has significant advantages for system efficiency.

6 Experimental Analysis: A BitTorrent Overlay Protocol

6.1 The BitTorrent Protocol

In this section, we discuss an application of our mechanisms to BitTorrent. In BitTorrent, the distributed work system is compromised of a collection of peers called a *swarm*. A swarm begins when a *seeder*, an altruistic peer that has a complete file, sets up a server and allows other peers to download the file. The file is partitioned into distinct pieces, and a unit of work in this system consists of the transmission of one piece from one peer to another. The peers in a swarm download pieces from the original seeder and share pieces with each other. A BitTorrent client 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 provide the highest upload rate in return, determined periodically, and the seeders assign their upload slots to those peers that 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* which is assigned via a 30 seconds round-robin shift over all the interested peers regardless of their upload rate. Due to optimistic unchoking, new peers have a chance to obtain their first pieces of data and bootstrap the process.

Up to a certain limit, the more bandwidth a peer gives, the more it gets in return, which provides downloaders in a single swarm with a strong incentive to upload to others. This policy is also often called "tit-for-tat" (even though Levin et al. [19] show that they are not formally equivalent). This mechanism was one of the crucial design choices of the BitTorrent protocol [7], providing much better incentives to the peers in the system than previous protocols like Gnutella [1].

However, these incentives are only temporary and local: the incentives only work in a bilateral way,

and there is no incentive to continue sharing the file after the download has finished. Ironically, it is even disadvantageous to share upon completing a file, 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.

6.2 Accounting Mechanisms for BitTorrent

When an accounting mechanism is available in a P2P file sharing system such as BitTorrent, this raises the question as to which allocation policy to use. A natural candidate would be the ranking policy, which always gives preference to agents with a higher score and successfully separates sharers from free-riders in the round-based simulations. 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 clients that are not using the accounting mechanism. Secondly, a user who installs a new client should have performance at least as good as with the old BitTorrent client.

This puts some restrictions on what kind of allocation policies we can usefully employ. Imagine agent i using an accounting-based client in a 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 ranking policy 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 could be significantly worse off using an accounting-based client. Consequentially, the ranking policy that we employ in our experiments with BitTorrent only uses the accounting mechanism to decide to whom to allocate the optimistic unchoking slot.

The second allocation policy we employ is the banning policy, which we update there to the BitTorrent domain in the following way. First, an agent never uploads to another agent with a score below a certain threshold. But aside from this banning operation, the policy uses the standard BitTorrent policy: the optimistic unchoking slot is assigned randomly to one of the peers that are not banned and the remaining slots are allocated to non-banned peers who provide the highest upload rate in return. The idea here is that the threshold is set in such a way that free-riders are banned, but cooperative agents will never reach a score lower than this threshold. Note that, similar to using the ranking policy for the allocation of all upload slots, it may also be sub-optimal for an agent to follow the banning policy. This is because an agent with a score below the threshold may reciprocate with the highest upload rate in the *current* swarm. However, there is an important difference between the ranking policy and the banning policy. As the size of a swarm grows, and thus the number of cooperative peers with high upload rates grows as well, the disadvantage to a client using the banning policy diminishes. Imagine there are 1,000 peers in the swarm, and the agent currently bans 10% of those, i.e., 100. This still leaves 900 peers from whom the agent will now select those who reciprocate with the most bandwidth. This is in contrast to employing the ranking policy for all slots, where swarm size does not matter. The ranking policy will always select the agents with the highest scores, even if they currently reciprocate with the lowest speed in the whole swarm.

Note that if a significant number of agents in the swarm use the accounting mechanism, then

there is also an upside for an agent to upload to an agent who also uses an accounting mechanism-based client, because the agent can expect to be rewarded (directly or indirectly) for this cooperative behavior in the future. Thus, whether and how much an agent would be worse off by employing the banning policy, would depend on the swarm size, the distribution of agent types in the swarm, and the particular banning threshold. In practice, a small disadvantage may be tolerable (most BitTorrent users do not use BitThief or BitTyrant even though they can be faster than standard BitTorrent clients), but a large disadvantage must be avoided. To guarantee that the disadvantage remains minimal, the particular banning threshold could be set dynamically by the client software, dependent on relative upload rates obtained from the different agents in the swarm. However, we do not address this particular aspect further in this work. Going forward, we focus on comparing the ranking and the banning policy. 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 the optimistic unchoking slot to the interested peers in order of their scores; the remaining slots are allocated to those peers who provide the highest upload rate in return.
- **Banning policy:** Peers do not assign *any* upload slots to peers that have a score which is below a certain threshold δ. The optimistic unchoking slot is assigned randomly to one of the peers that are not banned; the remaining slots are allocated to non-banned peers who provide the highest upload rate in return.

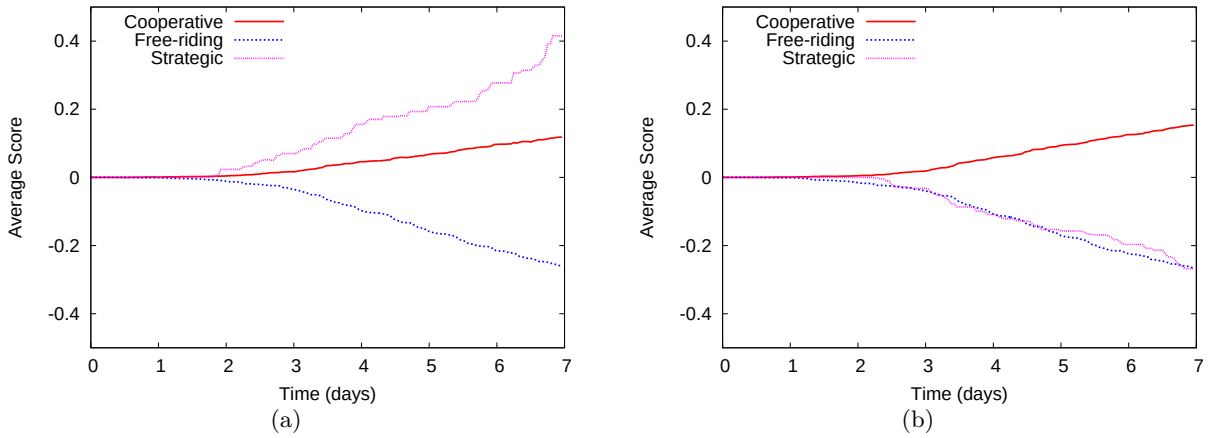
For some of our experiments, we consider an *Omniscient (Centralized Max-flow)* mechanism. To use this mechanism as a baseline, and to remove any misreport considerations, this mechanism stores the *true* up- and download statistics of peers in a central database, instead of in local databases. The scores are computed using the BASIC mechanism, based on the information stored in the central database.

In the following experiments, we want to identify which effects are due to centralized or decentralized information exchange, and which are due to misreport-proofness. To this end, we study the ranking and banning policies in combination with BARTERCAST-BASIC, BARTERCAST-DROP-EDGE, and the *Omniscient* mechanism.

6.3 Simulation Set-up

We have built a simulator which incorporates all relevant aspects of BarterCast and BitTorrent. We simulate an epidemic Peer Sampling Service using the decentralized BuddyCast protocol that is already implemented and released as part of the Tribler file sharing client [27]. Our simulator follows the BitTorrent protocol at the piece-level, including unchoking, optimistic unchoking, and rarest-first piece picking. We have combined all processes in a simulation environment that can generate workloads based on either probabilistic request arrivals or traces of data.

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 *cooperative*, a *free-riding*, or *strategic*. Free-riders immediately leave the swarm after finishing a download, while cooperative peers share every completed file for 10 hours. Strategic peers behave as free-riders, and in addition spread forged BarterCast messages in which they report a maximum upload to others and zero download. The strategic peers also make misreports when DROP-EDGE is being used, which doesn't benefit them, but which introduces additional noise into the system. All peers in our simulations have a 3 MBps downlink and a 512 KBps uplink, corresponding



(a) (b)

Figure 13: Evolution of the average scores over time for the three different agent types, using (a) BarterCast-Basic and (b) BarterCast-Drop-Edge.

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, the agents report their information about the 10 agents they have seen most recently, and the 10 agents with the highest upload to them.

6.4 Poisson-based Simulations

We generate a Poisson arrival process for file requests for each peer with an average of one request per day per peer. Requests are homogeneously distributed over the 10 files. Unless otherwise noted, all of the following results are based on simulations with 50% cooperative agents, 40% free-riders, and 10% strategic agents (different distributions of agents lead to qualitatively similar results). 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, we evaluate the influence of the banning threshold, and evaluate a simple model for behavioral change.

6.4.1 Evolution of Agents' Scores

As a first step, we study how the scores of the different agents evolve 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 *average score* $\mathbf{S}_i^M$ of agent i using mechanism M as the average of the scores that each of the other $N-1$ agents assign to i:

$$\mathbf{S}_i^M = \frac{1}{N-1} \sum_{j \neq i} S_{ji}^M(G_j, C_j) \tag{21}$$

In Figure 13 the evolution of the average scores over time are plotted for the the three different agent types for the two decentralized versions of the mechanisms. Figure 13 (a) shows the results for BarterCast-Basic. The scores of the free-riders clearly decrease over time, while the scores of the cooperative peers increase slightly. However, the strategic peers benefit a lot from manipulating the accounting mechanism; their scores are significantly higher than the scores of the other peers.

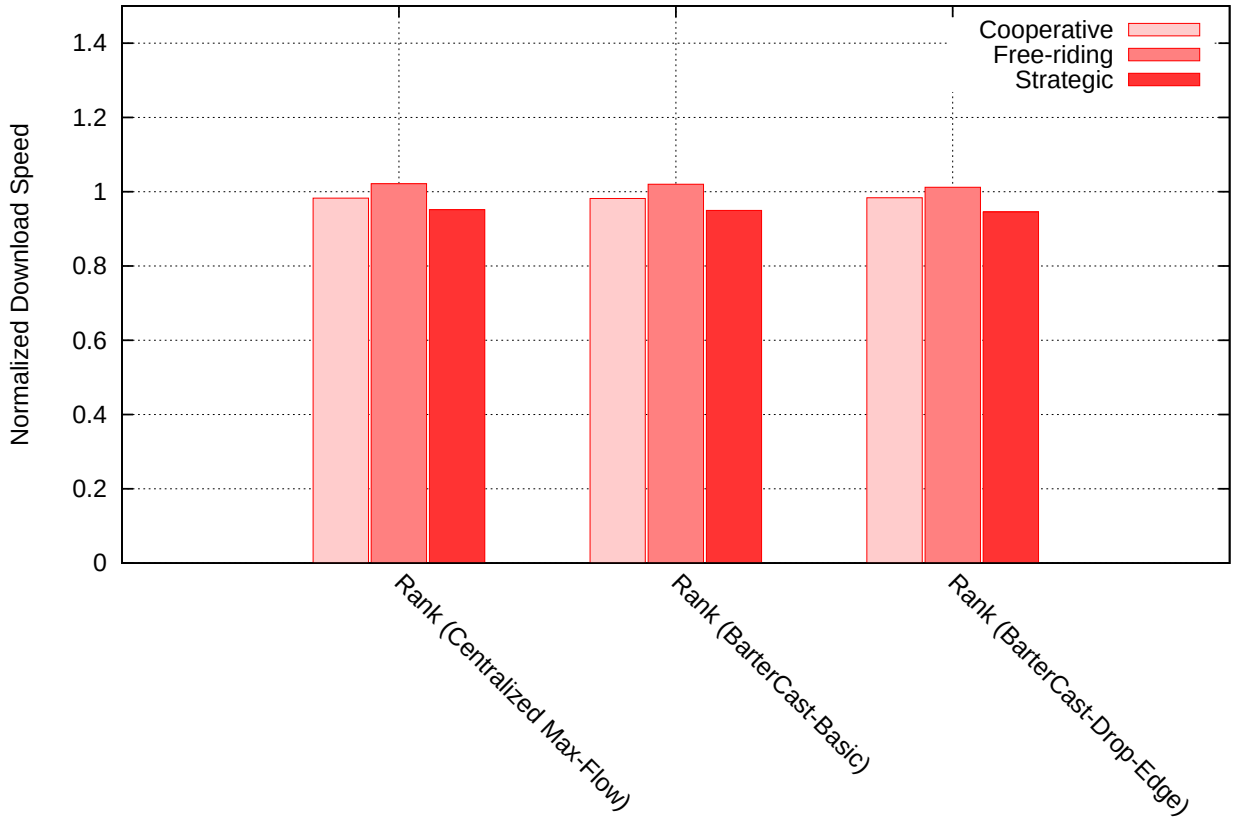


Figure 14: Comparison of the average download performance of cooperative agents, free-riders, and strategic peers for the ranking policy with different accounting mechanisms. The download speeds are normalized relative to the original BitTorrent mechanism (without accounting).

This effect vanishes completely when BarterCast-Drop-Edge is used, which is displayed in Figure 13 (b). Here, the strategic agents have the same average scores as the free-riders. Obviously, BarterCast-Drop-Edge is very successful in computing scores that separate cooperative agents from free-riders and strategic agents.

6.4.2 The Ranking Policy

In this section, we study study the effect of the ranking policy on the average download performance of the agents. All download speed results are normalized with respect to the results of simulations using the standard BitTorrent protocol without an accounting mechanism. We consider the centralized version of the Basic mechanism, as well as the two decentralized mechanisms BarterCast-Basic and BarterCast-Drop-Edge. In Figure 14, the normalized download speeds are plotted for cooperative, free-riding, and strategic agents, for the three different mechanisms under consideration. We see immediately that using any of the mechanisms, the ranking policy has no significant effect, i.e., the performance is virtually the same for all agent types.

An closer investigation of this effect shows that this is caused by the size of the swarms. As we simulate relatively small swarms, peers do not always have enough requests from other peers to fill all

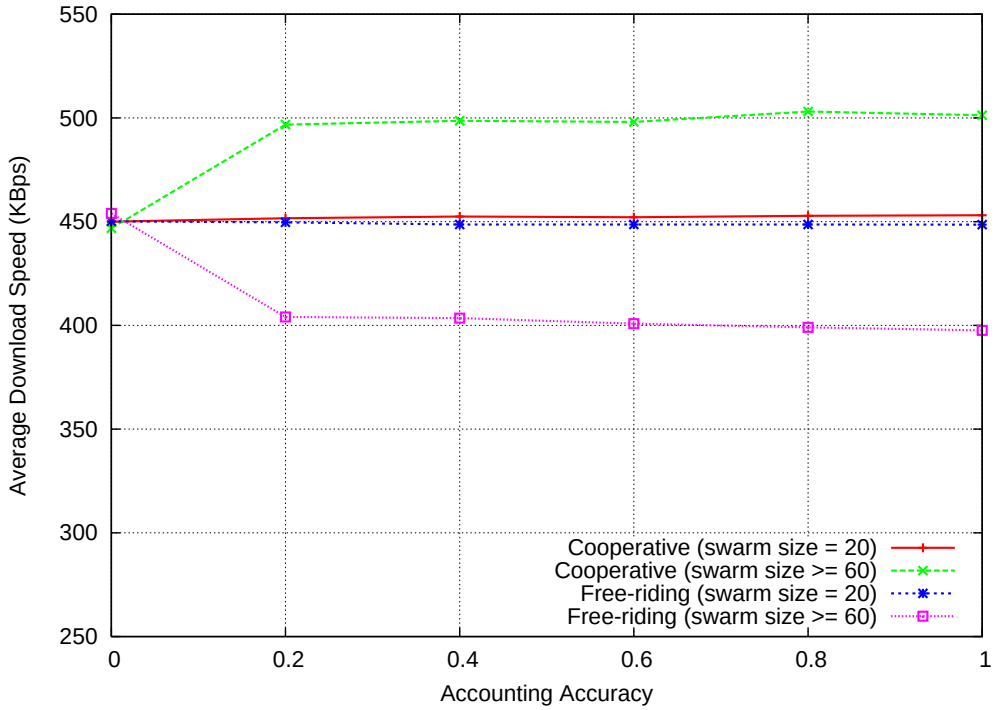


Figure 15: Performance of the Ranking Policy with different swarm sizes.

of their upload slots. Hence, free-riders and strategic agents can often find enough free slots to still have a normal performance. This suggests that the ranking policy can only be effective if swarms are relatively large *and* peers know a significant fraction of the other peers in the network. We argue that for the policy to be effective, it is necessary that a peer gets strictly more requests than he has upload slots, and that he has sufficient information about those peers to differentiate between cooperative agents and free-riders.

To verify this hypothesis, we ran additional experiments where we varied the swarm sizes between 20 and 180 and where we varied the accuracy of the information available to the agents. In Figure 15, we display the results from that experiment. On the x-axis we vary the *accounting accuracy*, where an accuracy of 0.4 means that each agent has accurate scores for 40% of the others agents in the network, and no information for the rest. First, it is easy to see that the performance plateaus at an accuracy of 20%. 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 small amount of information. Next, we consider the effect of varying the swarm sizes. With a very small swarm size of only 20 agents, the ranking policy is not effective in separating cooperative agents 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 swarms of some minimal size.

We also found that increasing the swarm size further from 60 to 180 caused no additional effect on the performance difference between cooperative agents and free-riders. It turns out that this effect occurs because when using the ranking policy, we only use the accounting mechanism to allocated the optimistic unchoking slot. Each agent has 5 upload slots, out of which 4 are allocated based on best

response rates. This explains why the performance of the cooperative agents compared to the free-riders 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 client are at least as well of as users of the standard BitTorrent clients, the maximal effect that employing the ranking policy can have is limited: we can at most reduce the performance of the free-riders by 20% compared to the cooperative agents. This may not be enough to incentivize free-riders to change their behavior and become cooperative. Thus, for the remainder of this section, we focus on the banning policy.

6.4.3 The Banning Policy

As in the previous section, all the performance results presented in this section are normalized with respect to the results of simulations that do not implement any policy at all. To study the banning policy in more detail, in particular to find good banning thresholds, we use the following monotonic function to transform the original scores (which can be between $-\infty$ and $+\infty$) such that they are all between -1 and $+1$:

Definition 22. ***(Normalized Scores)*** *Given subjective work graph G_i and choice set C_i, let G_i^B and G_i^D denote i's subjective work graphs after applying the* Basic *and the* Drop-Edge *mechanisms respectively. The normalized scores for an agent j are:*

$$S_{ij}^X(G_i, C_i) = \frac{\arctan(MF_{G_i^X}(j,i) - MF_{G_i^X}(i,j))}{\pi/2} \tag{22}$$

where $X \in \{B, D\}$.

For the remainder of this section, we will always use the normalized scores in our experiment. Note that an agent about whom no information is available will have a score of 0, and an agent who contributes as much work as he consumes will also have a score of 0, at least on average. Thus, for the banning policy to have an effect, the banning threshold must be set to a value between -1 and 0. For the first analysis we fix the banning threshold at $\delta = -0.5$; later we study the effect of varying the banning threshold between -1 and 0.

The results for using the banning policy in combination with the three different accounting mechanisms are shown in Figure 16. We see that the performance of both free-riders and strategic agents is almost the same in the omniscient (centralized max-flow) mechanism. The small difference is due to randomization in the simulation. But more importantly, we see that the free-riders and strategic agents achieve roughly half the performance of the cooperative agents. This parallels the results from the discrete, round-based simulations presented in Section 5.3, where we showed in Figure 12 that after a sufficient number of time steps, the efficiency of the cooperative agents was approaching twice the efficiency of the free-riders and strategic agents.

Next, we consider the two decentralized mechanisms. For BarterCast-Basic, the performance loss of the free-riders compared to the cooperative agents is again roughly 50%. However, now the strategic peers achieve a performance about twice as high as before, even higher than the cooperative peers. Of course, this is due to the misreport vulnerability of the Basic mechanism which the strategic peers exploit. Note that the cooperative agents' performance is a little bit lower than before, which can be explained by the fact that more of the bandwidth now goes to the strategic peers.

Finally, consider the BarterCast-Drop-Edge mechanism. Here, we see that the performance of the free-riders and the strategic peers is roughly the same again (the differences can only be

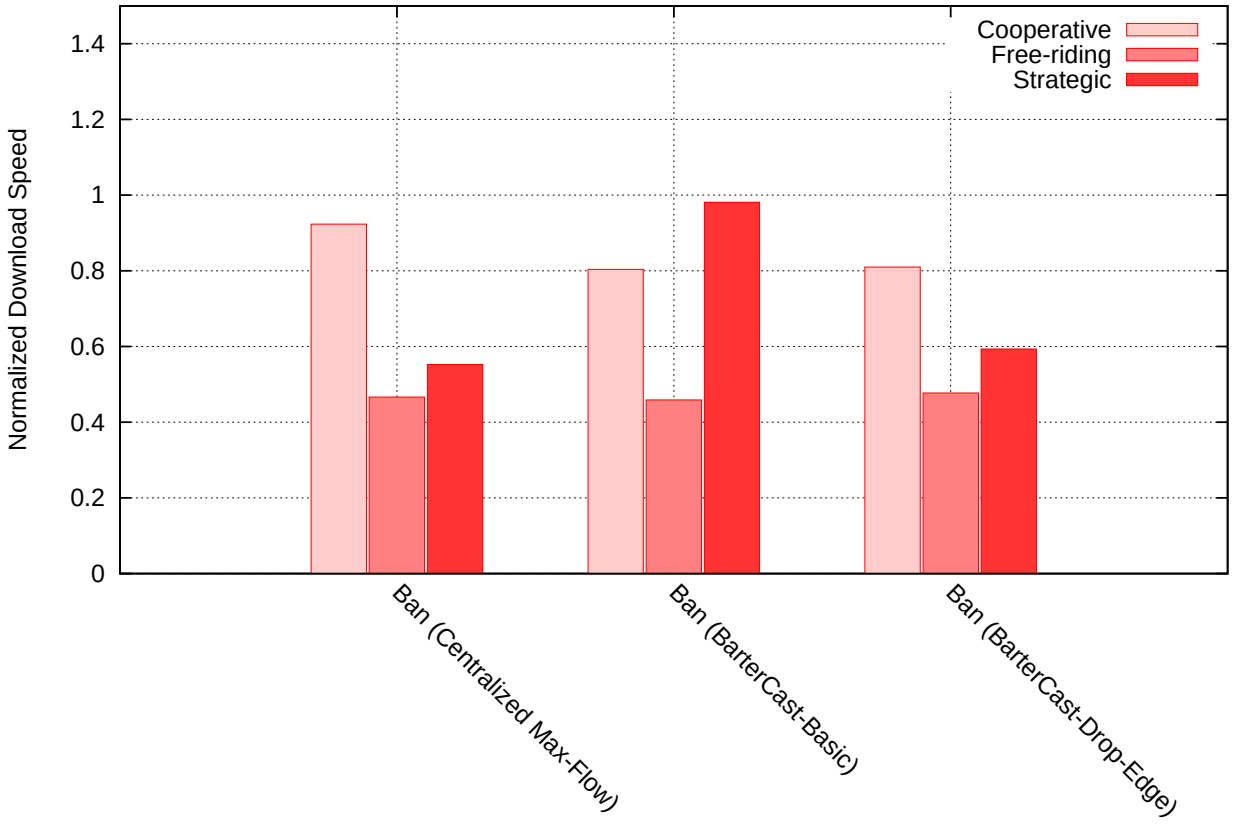


Figure 16: Comparison of the average download performance of cooperative agents, free-riders, and strategic peers for the banning policy with different accounting mechanisms.

attributed to random effects in the simulation), due to Drop-Edge being misreport-proof. Now the cooperative agents achieve the highest performance, about 30%-40% higher than the free-riders and strategic agents. Note that the performance of the cooperative agents under Drop-Edge is somewhat lower than under Centralized Max-Flow. This is due to the strategic agents, who spread false information even though they cannot benefit from it. This increases the overall noise in the system which leads to somewhat lower effectiveness of the accounting mechanism. Overall, we can conclude that using the banning policy with BarterCast-Drop-Edge is indeed effective in separating cooperative agents from free-riders and strategic peers. It achieves a larger performance difference than with the ranking policy. However, a performance difference of 30%-40% might still be too small to incentivize free-riders to become cooperative agents. Therefore, we study the effect of varying the banning threshold.

In Figure 17, the normalized download performance for cooperative, free-riding, and strategic agents is plotted for various thresholds using the banning policy with the BarterCast-Drop-Edge mechanism. As before, strategic agents have no benefit when the Drop-Edge mechanism is used, leading to performance comparable to that of the free-riders. The figure shows that the more strict the threshold (i.e., closer to 0), the larger the relative penalty for the free-riders and strategic agents compared to the cooperative agents. We see that we can easily achieve performance differences larger

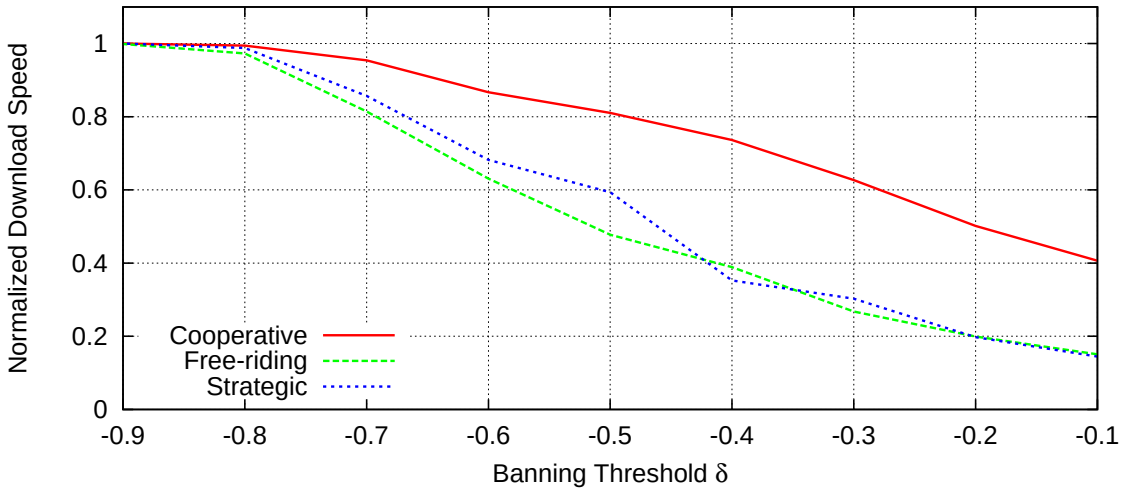


Figure 17: The normalized download performance of cooperative agents, free-riders, and strategic agents, using the banning policy for various thresholds δ under the BarterCast-Drop-Edge mechanism.

than 30%-40%. At $\delta = -0.1$, the relative performance difference is largest, with cooperative agents achieving roughly 2.5 times the performance of the free-riders. However, the absolute performance of the cooperative agents is also smallest at $\delta = -0.1$, which is clearly detrimental to our design goal. The lower performance of the cooperative agents is due to two effects. First, at such a high banning threshold, agents will sometimes also ban cooperative agents because of the imperfect information due to the decentralized information exchange protocol. Second, while free-riders and strategic agents try to exploit the system, they nevertheless do provide some bandwidth while they are still downloading files. If a large majority of the free-riders and strategic peers is banned, then their bandwidth is also lost from the cooperative agents' perspective.

This view, however, neglects the fact that in practice, we would expect a certain percentage of the free-riders and strategic agents to *change their behavior*, and become cooperative, if the banning policy with a high enough banning threshold were used. Thus, to find the optimal banning threshold, we must consider a behavior model.

6.4.4 Banning Policy with a Behavioral Change Model

As a system designer, setting the optimal banning threshold requires some assumptions regarding how free-riders and strategic agents will change their behavior when facing clients with that employ a banning policy. Assuming that many free-riders will become cooperative 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 former free-riders. However, if free-rider conversion is slow, the prolonged loss of performance of the cooperative agents might be unacceptable, and a milder threshold should be considered. To better understand this trade-off, we performed simulations assuming an illustrative relationship between the banning threshold δ and the percentage of free-riders and strategic agents in the system f:

$$f(\delta) = 0.8 \cdot (\frac{1}{16})^{\delta+1}. \tag{23}$$

With the above relationship, a system with no banning (i.e., $\delta = -1$) has 80% free-riders, while

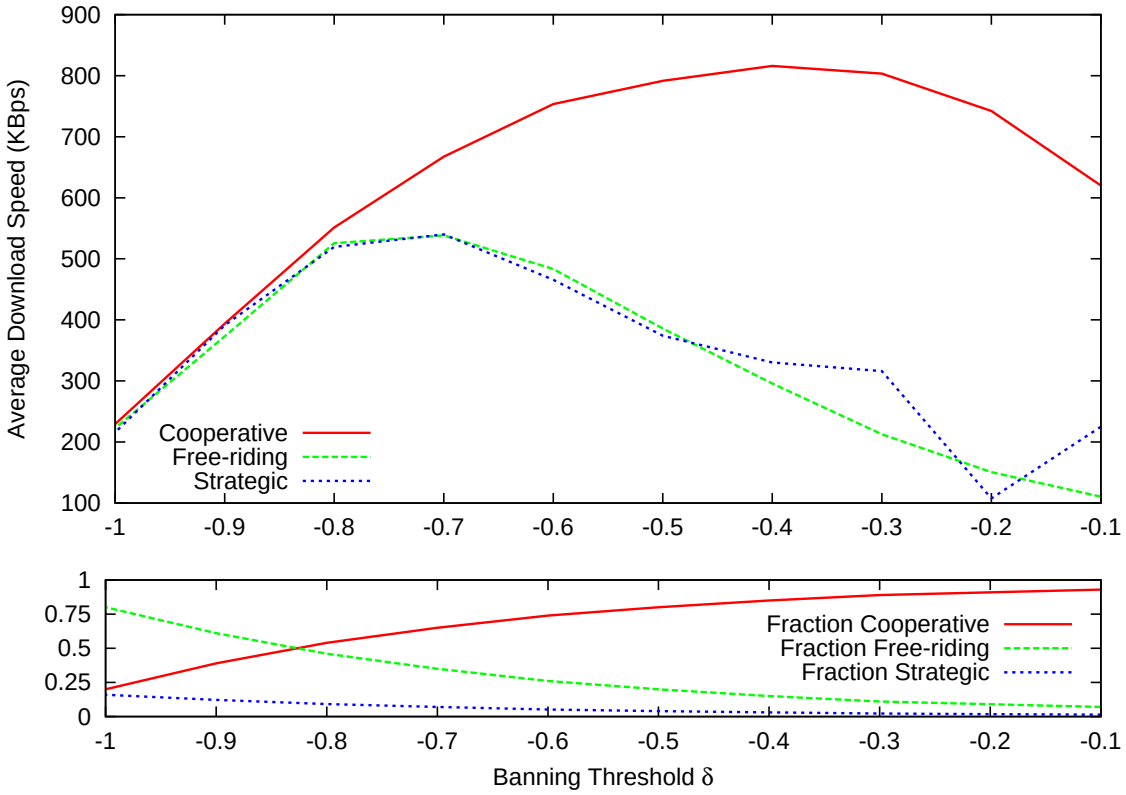


Figure 18: The performance of cooperative agents, free-riders, and strategic agents, assuming a behavioral model where more free-riders and strategic agents become cooperative as we increase the banning threshold for the BarterCast-Drop-Edge accounting mechanism. The bottom graph displays the distribution of the agent types corresponding to the different banning thresholds.

a system with very strict banning (i.e., $\delta = 0$) has only 5% free-riders. We assume that 25% of the free-riders are strategic. In Figure 18, we display the download speed for cooperative, free-riding, and strategic agents in a system with the above relationship. We observe that when the banning threshold is very low, all peers have a relatively low performance. This is intuitive, because there is hardly any penalty for free-riders, and thus many peers will freer-ride, which leads to little supply of resources. As we increase the banning threshold, the performance of all peers increases, as more and more of the free-riders and strategic peers become cooperative. At some point ($\delta > -0.8$), there are enough cooperative agents in the system for the banning of free-riders to become effective. Around $\delta = -0.4$, the download speed of the cooperative agents peaks, while the penalty for free-riding is very strong. As we increase the banning threshold further towards 0, the disadvantages from banning more free-riders, and sometimes banning even cooperative agents due to incomplete information, starts having a negative effect on the performance of the cooperative agents. Thus, the trade-off between sufficient banning of free-riders versus reducing unnecessary loss of performance for cooperative agents is clearly visible. 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.

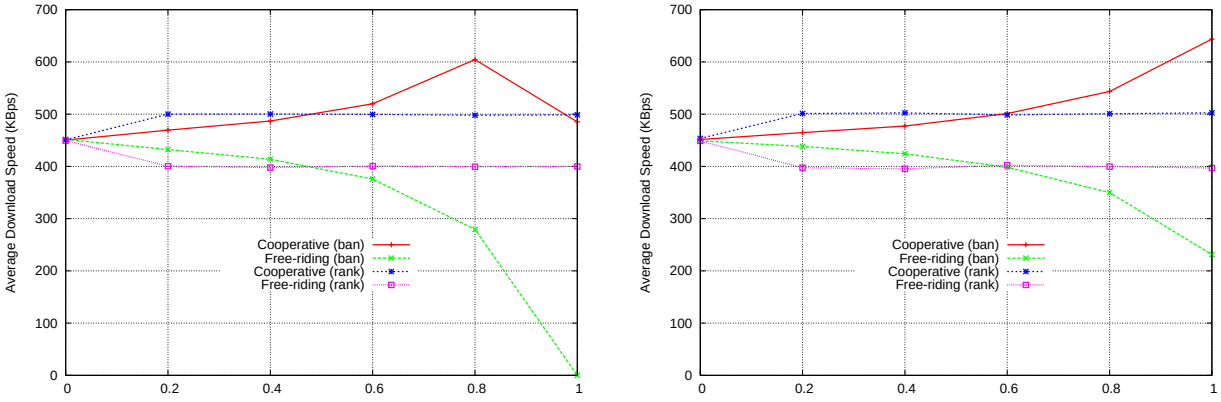
6.4.5 The Effect of Accounting Accuracy and Noise

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. However, we cannot simulate significantly larger BitTorrent networks with such detail 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. While the basic principle does not change when the network size increases from 100 to thousands or millions of agents, there are two aspects that do change. First, in very large networks, it is more likely that two agents that meet have little or no information about each other (i.e., are disconnected in the work graph). Second, the information that *is* available to the agents may be very noisy because the decentralized information exchange protocol needs a long time to spread information through a large network, and using the max-flow algorithm further distorts the scores. We seek to better understand these two challenges.

In contrast to all experiments we have presented so far, in the experiment we discuss here, the agents do not *compute* the scores of the other agents themselves. Instead, we inject the scores, giving us the ability to control precisely how much and which information each agent has available when making a decision. 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 of using max-flow is that the scores computed by max-flow are only approximations for the net work performed by an agent. In our experiments, we model this *noise* as the variance of the distribution from which we draw an agent's view of another agent's score.

In Figure 19 we show 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). We draw the agents' scores from a gaussian distribution with mean equal to the true scores, and with a standard deviation equal to 0.0 (no noise, i.e., perfect information), 0.2, 0.4 and 0.8, which corresponds to the four graphs (a)-(d). 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 cooperative agents from free-riders, giving them a performance ratio of 5 to 4, and this stays the same even as we increase the accuracy to 1.0. We have



(a) Noise = 0.0 (b) Noise = 0.2

(c) Noise = 0.8 (d) Noise = 2.0

Figure 19: Analyzing the effect of accounting accuracy and noise on the banning policy (with threshold $\delta = -0.5$) and the ranking policy. Here, no accounting mechanism is used. The agents make allocation decisions using scores that are drawn from a gaussian distribution, with mean equal to the true scores, and standard deviation equal to the noise value.

already explained in Section 6.4.2 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 19 (c)). The explanation is simple: the ranking policy picks the agent with highest scores, and of course the average scores of the cooperative agents are much higher than the average scores of the free-riders. Thus, with some noise in the system, some highest-ranked cooperative agents might change their relative rank, but it is still very likely that the highest ranked agent (out of all agents) in any given choice set will be a cooperative agent and not a free-rider. Only when we increase the noise even further does the result change a little bit, i.e., the performance for cooperative agents decreases slightly and the performance of the free-riders increases slightly. However, such high values of noise are not realistic in practice.

Now we turn our attention to the *banning policy*, where we used a banning threshold of $\delta = -0.5$. As we see in Figure 19, the effects of accounting accuracy and noise are much more pronounced. In particular, the performance difference between cooperative agents and free-riders keeps increasing as we increase the accounting accuracy from 0 to 1. This is expected because every agent that another agent has no information about has a score of 0, and thus will not be banned. Finally, we 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 cooperative agents and free-riders decreases significantly. For example, with accuracy 0.8 and noise level 0, cooperative agents achieve 600KBps and free-riders achieve 300KBps.[9] For an accuracy of 0.8 and noise level of 0.2, the cooperative agents' performance drops to 550KBps, and the free-riders' performance increases to 350KBps. For a noise level of 0.8, the performance difference achieved via the banning policy is smaller than with the ranking policy, except for very high accuracy values, where the two policies perform essentially equally well. This decrease in the effectiveness of the accounting mechanism with banning is expected. Remember that the banning threshold is set in such a way that free-riders are banned and cooperative agents are not, and fine-tuning the threshold will always involved a trade-off between banning too many cooperative agents and too few free-riders. Now, by adding noise to the system, two kinds of mistakes start to happen: not only are some of the free-riders not banned that should be banned, but also some of the cooperative agents are now banned that should not be banned. This is in contrast to the ranking policy, where adding noise at first only affects which of the cooperative agents gets the highest score, but it takes a lot more noise, until a free-rider makes it to the top. Furthermore, the frequency of mistakes is also higher for the banning policy than for the ranking policy, because the banning policy can potentially affect every agent that is considered for any upload slot, not just the agents being considered for the optimistic unchoking slot.

To conclude this analysis, we can make some assumptions regarding what accuracy and noise levels to expect in real BitTorrent systems. Piatek et al. [25] have shown empirically that 99% of BitTorrent peers are connected via paths of length 2. Thus, even using a max-flow algorithm restricted to 1 or 2 hops, we can expect an accounting accuracy of 0.99 in real BitTorrent networks. It is a little more difficult to estimate the noise level of the accounting scores. In our own experiments using the round-based simulations as well as the BitTorrent simulations, we found that even though a decentralized information exchange protocol is used and with max-flow further distorting the accounting scores, the BarterCast-Drop-Edge mechanism can differentiate between cooperative agents and free-riders, even after just a few time steps and when only a few BarterCast messages have been exchanged (compare Figure 12(b) and Figure 13(b)). This suggests that even in larger networks we can expect

[9] The performance drop of the cooperative agents for noise level 0 and accuracy 1.0 occurs because with perfect information, all free-riders are banned from the system, which are then also no longer available to do tit-for-tat with the cooperative agents. This is the same trade-off we discussed in Section 6.4.4.

a reasonably low level of noise. Based on these assumptions, considering Figure 19(b), we would expect the banning policy to cause a significant performance difference between cooperative agents and free-rider, even in large networks. However, evaluating our mechanisms and allocation policies on a larger scale, ideally with real users, remains a formidable research challenge.

7 Conclusion

In this paper, we have studied distributed work systems, where agents perform small units of work for each other, without the ability for a third party to monitor those bilateral interactions. The overall goal is to incentivize agents to be cooperative, i.e., to perform as much work as they consume, and to prevent free-riding. We have shown that previous approaches to solve this problem via *trust* or *reputation* mechanisms are not suitable, and propose to treat the problem as an *accounting* task instead. The Drop-Edge mechanism removes any incentive for the agents to misreport, by selectively dropping some of the information available to the agents when considering for whom to perform work. In our theoretical analysis, we have proved that the information loss of Drop-Edge is small and vanishes in the limit as the number of agents in the network grows. The second class of manipulations we have considered are sybil attacks. We have shown that under reasonable assumptions, no accounting mechanism can be sybil-proof. However, we have also shown that a weaker robustness property, K-sybil-proofness, can be achieved for a limited class of sybil attacks.

In the second part of the paper, we have coupled Drop-Edge with BarterCast, a decentralized information exchange protocol, to study how accounting mechanisms can be used to improve the efficiency in distributed work system. First, we have performed discrete, round-based simulations. Whereas manipulations are very useful without Drop-Edge, the Drop-Edge mechanism removes this problem and provides cooperative agents with higher efficiency, while free-riding and strategic agents have lower efficiency. In a second set of experiments, we have tested the effectiveness of accounting mechanisms as an overlay protocol for BitTorrent. Using Tribler, a real P2P file sharing client that is already deployed and being used in practice, we were able to run simulations at the BitTorrent protocol level. We have analyzed two different allocation policies, to decide how to allocate work based on the scores computed by the accounting mechanism: the ranking policy and the banning policy. The effectiveness of the ranking policy is limited in BitTorrent because we can only use it to allocate the optimistic unchoking slot. However, using the banning policy with a finely-tuned banning threshold, we can separate cooperative agents from free-riders, such that the performance of the cooperative agents 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 cooperative. Assuming such a behavioral change, we have demonstrated significantly improved system efficiency. 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 cooperative agents and free-riders to a large enough degree. Based on previous results and our own experiments, we expect that in large, real-world P2P file sharing networks, accounting accuracy would be relatively high noise levels would be relatively low, such that accounting mechanisms would be expected to be successful, even in very large networks with millions of agents. However, studying the effectiveness of accounting mechanisms in real-world systems, and in large networks, remains an exciting research challenge.

References

[1] Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[2] Noga Alon, Felix Fischer, Ariel D. Procaccia, and Moshe Tennenholtz. Sum of Us: Strategyproof Selection from the Selectors. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, Groningen, NL, July 2011.

[3] Nazareno Andrade, Miranda Mowbray, Aliandro Lima, Gustavo Wagner, and Matei Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems (P2PECON)*, New York, NY, 2005. ISBN 1-59593-026-4.

[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*, Toronto, Canada, May 2010.

[5] Alice Cheng and Eric Friedman. Sybilproof Reputation Mechanisms. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, pages 128–132, Philadelphia, PA, August 2005.

[6] Alice Cheng and Eric Friedman. Manipulability of PageRank under Sybil Strategies. In *Proceedings of the 1st Workshop of Networked Systems (NetEcon06)*, Ann Arbor, MI, June 2006.

[7] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, Berkeley, CA, June 2003.

[8] Pranav Dandekar, Ashish Goel, Ramesh Govindan, and Ian Post. Liquidity in Credit Networks: A Little Trust Goes a Long Way. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, San Jose, CA, June 2011.

[9] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, New York, NY, May 2004.

[10] 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, 2006.

[11] Eric Friedman, Paul Resnick, and Rahul Sami. Manipulation-Resistant Reputation Systems. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, pages 677–698. Cambridge University Press, New York, NY, 2007.

[12] 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.

[13] Andrew V. Goldberg and Robert E. Tarjan. A New Approach to the Maximum-Flow Problem. *Journal of the ACM*, 35(4):921–940, 1988.

[14] Philippe Golle, Kevin Leyton-Brown, Ilya Mironov, and Mark Lillibridge. Incentives for Sharing in Peer-to-Peer Networks. *Electronic Commerce*, 2232:75–87, 2001.

[15] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A Reputation System for Peer-to-Peer Networks. In *Proceedings of the 13th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Monterey, CA, June 2003.

[16] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, Budapest, Hungary, May 2003.

[17] Ian A. Kash, Eric J. Friedman, and Joseph Y. Halpern. Optimizing Scrip Systems: Efficiency, Crashes, Hoarders, and Altruists. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, San Diego, California, June 2007.

[18] Ramayya Krishnan, Michael D. Smith, Zhulei Tang, and Rahul Telang. The Virtual Commons: Why Free-Riding can be Tolerated in Peer-to-Peer Networks. In *Proceedings of the Workshop on Information Systems and Economics (WISE)*, Seattle, WA, December 2003.

[19] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. BitTorrent is an Auction: Analyzing and Improving BitTorrents Incentives. In *Proceedings of the ACM SIGCOMM Conference*, Seattle, WA, August 2008.

[20] Qiao Lian, Yu Peng, Mao Yang, Zheng Zhang, Yafei Dai, and Xiaoming Li. Robust Incentives via Multi-level Tit-for-Tat. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, February 2006.

[21] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free Riding in BitTorrent is Cheap. In *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets)*, Irvine, California, November 2006.

[22] Michel Meulpolder, Johan Pouwelse, Dick H. Epema, and Henk J. 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.

[23] Michel Meulpolder, Lucia D'Acunto, Mihai Capotă, Maciej Wojciechowski, Johan A. Pouwelse, Dick H. J. Epema, and Henk J. Sips. Public and Private BitTorrent Communities: A Measurement Study. In *Proceedings of the 9th International Conference on Peer-to-peer Systems (IPTPS)*, San Jose, CA, April 2010.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[25] 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.

[26] Johan A. Pouwelse, Pawel Garbacki, Dick H. J. Epema, and Henk J. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, Ithaca, NY, February 2005.

[27] Johan A. Pouwelse, Pawel Garbacki, Jun Wang, Arno Bakker, Jie Yang, Alexandru Iosup, Dick H. J. Epema, Marcel J. T. Reinders, Maarten R. van Steen, and Henk J. Sips. Tribler: A Social-based Peer-to-peer System. *Concurrency and Computation: Practice and Experience*, 20(2): 127–138, 2008.

[28] Paul Resnick and Rahul Sami. Sybilproof Transitive Trust Protocols. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, Stanford, CA, July 2009.

[29] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN)*, San Jose, CA, January 2002.

[30] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems*, 9(2):170–184, 2003.

[31] 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.

[32] Dan Sheldon and John Hopcroft. Manipulation-Resistant Reputations Using Hitting Time. In *Proceedings of the 5th Workshop on Algorithms for the Web-Graph (WAW)*, San Diesgo, December 2007.

[33] 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.

[34] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin G. Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, Berkeley, CA, June 2003.

[35] Matthew Webb, Yu Mengquan, and Mark Beach. Propagation Characteristics, Metrics, and Statistics for Virtual MIMO Performance in a Measured Outdoor Cell. *IEEE Transactions on Antennas and Propagation*, 59(1):236–244, 2011.

[36] Chao Zhang, Prithula Dhungel, Di Wu, Zhengye Liu, and Keith W. Ross. BitTorrent Darknets. In *Proceedings of the 29th Conference on Information Communications (IEEE INFOCOM)*, San Diego, CA, March 2010.