BarterCast
Fully Distributed Sharing-Ratio Enforcement in BitTorrent

Meulpolder, Michel; Pouwelse, Johan; Epema, Dick; Sips, Henk

Publication date
2008

Citation (APA)
Meulpolder, M., Pouwelse, J., Epema, D., & Sips, H. (2008). *BarterCast: Fully Distributed Sharing-Ratio Enforcement in BitTorrent*. Delft University of Technology.

Delft University of Technology
Parallel and Distributed Systems Report Series

BarterCast: Fully Distributed Sharing-Ratio Enforcement in BitTorrent

Michel Meulpolder, Johan Pouwelse, Dick Epema, and Henk Sips
{M.Meulpolder, J.A.Pouwelse, D.H.J.Epema, H.J.Sips}@tudelft.nl

report number PDS-2008-002

ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Section
Faculty of Information Technology and Systems Department of Technical Mathematics and Informatics
Delft University of Technology
Zuidplantsoen 4
2628 BZ Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
`reports@pds.twi.tudelft.nl`

Information about Parallel and Distributed Systems Section:
`http://pds.twi.tudelft.nl/`

Abstract

Peer-to-peer (P2P) file sharing systems rely on the sharing of resources by large fractions of participants. Sharing-ratio enforcement provides a very strong incentive for peers to contribute, leading to a higher performance for all peers in the system. One way of implementing this is by banning peers that do not share enough, determined by their past behavior (i.e., *reputation*). Various reputation mechanisms have been designed to facilitate this, but they are centralized or not feasible in practice. In this paper, we present a secure, fully distributed mechanism for reputation management and its integration with BitTorrent. The resulting system enforces a long-term balanced sharing-ratio for all peers in a BitTorrent file sharing network. In our system, up- and download statistics are spread among peers and used to compute a subjective reputation for each peer. We apply the maxflow algorithm to limit the effect of peers that spread false information. We present simulations that demonstrate the system's accuracy and effectiveness. The resulting system overcomes the long-standing problem of BitTorrent's lack of seeding incentives, without any need for centralized administration, authority, or technology.

Contents

1 Introduction **4**

2 The BarterCast protocol **5**
2.1 Statistics exchange . . . 5
2.2 Local view of the network . . . 6
2.3 Reputation metric . . . 6
2.4 Spreading false information . . . 7

3 Reputation-based BitTorrent **7**
3.1 BitTorrent Tit-for-Tat . . . 8
3.2 Reputation-based unchoking . . . 9
3.3 Short-term versus long-term behavior . . . 9

4 Results **9**
4.1 Simulation setup . . . 9
4.2 BarterCast distributed reputation . . . 10
4.3 Reputation-based BitTorrent . . . 10

5 Related work **11**

6 Conclusions and future work **12**

List of Figures

1 Snapshot of a network where both data and BarterCast messages have been exchanged. The thickness of the data lines indicates the amount of data that has been transferred. 5
2 The reputation vs. the net contribution of a peer. 7
3 An illustration of three cases where peer B provides false (exaggerated) information to peer A in an attempt to gain a high reputation. 8
4 Accuracy of the reputation system; the difference between the objective and the subjective rank for all peers (top) and a histogram of this difference (bottom). In the top plot peers are sorted according to their objective rank. 10
5 The average reputation over time for sharers and freeriders in a system with 25% freeriders. . 11
6 The relative speed of freeriders over time. 11
7 The average download speed of sharers and freeriders using normal BitTorrent and reputation-based BitTorrent with different values of γ and 25% freeriders. 12

List of Tables

1 Introduction

File sharing systems are only successful when a significant amount of the users is willing to donate resources and to give honest information about their content. Ever since the first significant file sharing systems such as Napster and Kazaa, attempts have been made to keep track of the reputation of users in order to provide *sharing-ratio enforcement*, i.e., ensuring that a peer uploads to the network a given fraction of what it downloads from it. In the original Kazaa system, the client itself was responsible for broadcasting its own reputation value. Not surprisingly, this was very soon exploited by the *Kazaa Light* client which by default broadcasted a very high reputation value, thereby undermining the benefits of the reputation mechanism that all could have enjoyed. More secure distributed mechanisms provide theoretically valid frameworks, but are often not feasible in practice and none of these are successfully deployed. In today's file sharing world, networks have very large, dynamic user populations with asymmetry of interest, low rendez-vous probability and high churn (i.e., lots of peer arrivals and departures in a short time) [**?**]. The well-known EigenTrust [**?**] algorithm provides a globally consistent objective trust value for all peers in a network but is computationally heavy and not suitable for networks with high churn. The currency-based MojoNation system was deployed but failed to work in practice due to economic effects as described in [**?**]. Other alternatives such as [**?**, **?**] provide theoretical mechanisms that rely partly on centralized architecture.

With a lack of secure, practically feasible distributed mechanisms, many modern networks still rely on centralized reputation management. Popular closed BitTorrent trackers such as `www.TVTorrents.com` and former `oink.cd` require a minimum sharing-ratio (upload vs. download) for a user to stay in the system, and ban users that upload content of low quality or spam. In such systems, users can create an account and get a certain *grace period* (e.g., one month) in which they can download even though their sharing-ratio is very low. They have to utilize this period to upload as much as possible. After the grace period a minimum sharing-ratio is necessary to be able to download. Peers that do not meet the sharing-ratio requirements are simply banned from the network without mercy.

When using these systems, it is clear on first sight that the performance, reliability, and content quality are superior to systems without such rules. However, centralized management creates a central point of authority which is vulnerable to both technical and human failure. It is our goal to provide *fully distributed* P2P technology that provides the benefits and higher performance induced by sharing-ratio enforcement while being decentralized and secure. We present our fully distributed mechanism for reputation management, BARTERCAST, which is deployed in the open source file sharing network Tribler [**?**, **?**]. In BarterCast, real-time upload and download statistics are broadcasted to peers that are encountered with an epidemic protocol. Peers compute each others reputation based on their local view of direct and indirect traffic information. The spreading of false information is limited by applying the maxflow algorithm in computing indirect contributions of one peer to another peer. Furthermore, we present an adaptation of the BitTorrent protocol which uses the reputation information provided by BarterCast to create fully distributed sharing-ratio enforcement in BitTorrent transactions. By implementing an unchoking policy that also takes reputation into account, the earlier mentioned effect of central sharing-ratio enforcement can effectively be reached without any central component and free-riders are gradually banished from the system. The resulting approach overcomes the long-standing problem in BitTorrent that there are no long-term incentives to seed.

We present the results of extensive simulations of our approach, based on real traces of the `filelist.org` BitTorrent tracker. With this tracker it is possible to trace the behavior of unique peers over multiple sessions and multiple swarms, which is impossible with public trackers such as `mininova.org`. We show that the subjective, distributed reputation is consistent with the real behavior of peers and that the reputation of freeriders gradually decreases when they do not seed their finished downloads. We show that even in simulations of one week of activity, the download speed of freeriders has already decreased to a mere 50% of that of the sharing peers, therefore giving them a strong incentive to seed. We analyze the strictness of the policy as well as the effect of different fractions of freeriders, and conclude that our system successfully generates the effect of sharing-ratio enforcement that until now has only been implemented with centralized technology and authority.

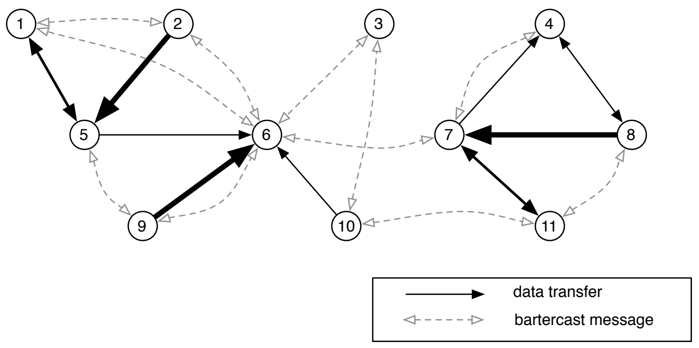

Figure 1: Snapshot of a network where both data and BarterCast messages have been exchanged. The thickness of the data lines indicates the amount of data that has been transferred.

2 The BarterCast protocol

In this section we present the BarterCast protocol and the way it can be used in BitTorrent. The outline of the protocol is as follows. First, up- and download statistics are spread in the network among peers. Second, these statistics are used by a peer to create a local view of the data transfer in the network. Third, based on this local view a peer computes the reputation of other peers it encounters. In the next section we will use this reputation in the BitTorrent protocol to create incentives for peers to share. We will describe first describe the BarterCast protocol in more detail.

2.1 Statistics exchange

In BarterCast, an epidemic protocol is used for peer discovery. Peers exchange lists of random peers and taste buddies with all peers they encounter. This epidemic protocol is the same as in [**?**]. Peers that have been discovered are from then on periodically polled to check their connectability and to exchange new peerlists.

In addition, every peer keeps track of the amount of data it has exchanged with any other peer during BitTorrent bartering sessions. These statistics are then periodically exchanged with all peers that are known via the peer discovery protocol. This is done by exchanging a *BarterCast message* which contains a selection of n peers and the amount of KBs up- and downloaded per peer. In Figure 1 a snapshot of an example network is shown where some of the peers have exchanged data with BitTorrent and some have exchanged BarterCast messages.

In this way, a peer accumulates two types of up- and download statistics: (1) statistics based on its own bartering with other peers; (2) statistics received from a known peer, based on the bartering of this known peer with a third party. It is important to note that a peer only spreads statistics of the first type to others. For example, in Figure 1, peer 6 reports to others about the number of KBs it exchanged with 5, 9, and 10. It also knows (from peer 2's BarterCast message) how much peer 2 uploaded to peer 5, but it will not spread this information any further.

In the current implementation the number of peers in a message is limited to 10, being the top 10 peers that have uploaded the most to the peer sending the message. We leave the analysis of the message size and overhead to future work, but remark here that in our experiments and deployment the current message overhead was negligible compared to the other data transfers between peers.

2.2 Local view of the network

As a result of the message exchange, each peer accumulates its own direct information and the information received from others. With this information, a peer builds a graph of all peers it knows, with pointed edges that represent data transferred from one peer to another peer (such as in Figure 1). Note that not all peers have to be (indirectly) connected with the peer itself.

Since there might be peers that lie or collude to report false information, any second-hand information can not be trusted a-priori. In order to limit the effect of false information, we base our reputation mechanism on the *maxflow* algorithm. The maxflow algorithm computes the maximum 'flow' between two nodes in a graph over all possible paths. In our case, the graph consist of up- and download transfers between peers. In this way, a peer will be very limited in exaggerating its own contribution in the network (see Section 2.4 for a more detailed discussion of this property).

In [**?**], the maxflow approach is discussed as an approach to prevent colluders in P2P networks, however, this paper does not discuss how to actually distribute the necessary information, how to provide a solid reputation metric, and how to integrate the technique with an existing P2P protocol such as BitTorrent. Moreover, this paper does not provide extensive simulation results based on an actual distributed implementation and real network traces.

For the actual maxflow computation, we use the popular *Ford-Fulkerson* algorithm [**?**]. It works by repeatedly finding a path from a start node to a given end node over which there is a flow (e.g., data transfer), until no remaining path with a flow is left. The complexity of the algorithm is bound by a given limit of the path length. In our case, a peer can therefore compute the flow between any peer that it knows and itself. Between two peers that are not directly or indirectly connected the flow is naturally zero; they will therefore regard each other as neutral. Note that the network information a peer has is not necessarily complete. However, the aim is to provide a subjective reputation value that approximates and represents the real behavior of a peer. In our experiments we will show that despite the partial information, this approximation is accurate enough to realize the desired sharing-ratio enforcement.

2.3 Reputation metric

The maxflow algorithm provides an indication of the *upload* and *download* of a peer to/from another peer. In our fully distributed approach, we cannot use the sharing-ratio (*upload* : *download*) as a reputation value like central trackers do, since with this metric a new peer cannot be distinguished from a 'bad' peer, i.e., both have a very small upload and therefore a small, positive reputation value. Central trackers cope with this simply by granting a grace period in which the user is free to bootstrap its sharing-ratio. Since we do not have central management, we need another way to distinguish good, bad, and neutral peers. We therefore use (*upload minus download*) as a base, so that good peers have a positive value, bad peers have a *negative* value, and neutral peers have a value around zero.

We compute the subjective reputation of peer j at peer i using the values computed by on the maxflow-algorithm [**?**]. We denote $f_{i \to j}$ as the maxflow from peer i to peer j. A constant γ is used to scale the relationship between the reputation value and the flow. The reputation $R_i(j)$ of peer j at peer i is then computed as follows:

$$R_i(j) = \frac{\arctan(\gamma(f_{j \to i} - f_{i \to j}))}{\pi/2}$$

Positive values of $R_i(j)$ indicate that the upload flow from peer j to peer i is higher than the download flow, i.e., peer j has contributed more to peer i than what it has taken from it. Negative values in the same way indicate the opposite. The resulting reputation value is scaled between -1 and 1 (See Figure 2). The larger the flow of data, the higher the (absolute) value. As a result of the arctan function, the difference between 0 and 100 MB is more significant than the difference between 1 GB and 1.1 GB.

Note that the resulting reputation is subjective, i.e., a peer can have a high reputation at one peer and a low reputation at another. When a peer uploads a lot to many peers over time, its reputation will be

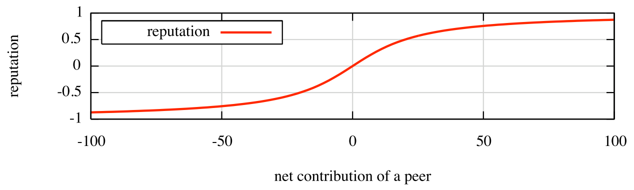

Figure 2: The reputation vs. the net contribution of a peer.

high with many peers in the network. We therefore have an ego-centric view of reputation without enforcing global consistency. This resembles the subjective, social concept of reputation in the real world.

2.4 Spreading false information

As shown in the previous section, the reputation of a peer B in the eyes of another peer A is based on its direct and indirect contribution to peer A. This contribution is computed using the maxflow algorithm over all possible paths from B to A. Naturally, the maximum flow over each path in the network is bound by the edge with the smallest flow. Since peer B sends messages containing his own *direct* contributions to others, the peer can lie and exaggerate this contribution with the aim of getting a high reputation. However, due to the application of the maxflow algorithm, such an exaggeration can only have a very limited effect. This is due to the following:

1. If B exaggerates a *direct* upload to A, the lie has no effect since A itself knows exactly what it has received from B.
2. If B exaggerates an upload to some peer C, which lies on a path to A, the maximum flow over this path will still be bound by other values on this path. These values can not be spread by peer A.
3. If B would collude with other peers that report an exaggerated value to create a path to A with a high flow, the maxflow is still bound by the actual contribution of the last peer to A. There, point (1) holds again.

Figure 3 illustrates these three cases. Note that in case (3), the *actual* maxflow from B to A is 5, but the information A has is as follows; it is claimed via bartercast that B uploaded 1000 to C and C uploaded 1000 to D, but since A only received 15 from D this will be the maxflow it computes. Even though this is slightly higher than the real value in this case, the large exaggeration of 1000 has not nearly the effect desired by B and C. Summarizing, a peer can only gain a high reputation at another peer if it actually uploads a lot to this peer or to peers already having a high reputation with this peer.

3 Reputation-based BitTorrent

In this section we will discuss our application of the BarterCast reputation system to prevent freeriding in BitTorrent. BitTorrent currently only implements a Tit-for-Tat policy which gives downloaders in a single swarm an incentive to upload to each other. However, there is no incentive to continue sharing the file after the download has finished. Ironically, it is even disadvantageous to share, since the consumed upload bandwidth cannot be used to do Tit-for-Tat in other downloads, which makes these downloads slower. BitTorrent therefore clearly suffers from the *Tragedy of the Commons* [**?**]. In our system however, freeriding is slower than more altruistic behavior. We will first give a short overview of the BitTorrent Tit-for-Tat policy, and after describe our adaptation of this policy to our reputation system.

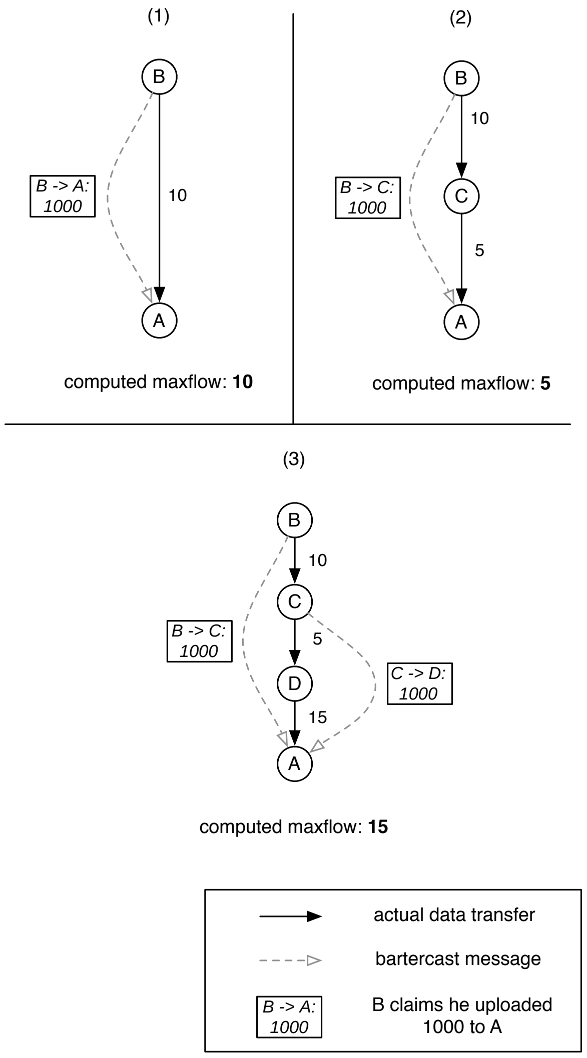

Figure 3: An illustration of three cases where peer B provides false (exaggerated) information to peer A in an attempt to gain a high reputation.

3.1 BitTorrent Tit-for-Tat

The normal BitTorrent protocol maintains a limited number of simultaneous upload slots (usually 4-7 depending on the implementation). Peers that do not yet have the complete file (*leechers*), assign their slots to those peers that currently provide the highest upload rate in return, determined periodically. Peers that have the complete file (*seeders*) assign their upload slots to those peers that have the fastest download rate. Peers that get a slot are called *unchoked*, while the other peers are *choked*. Furthermore, there is one extra slot for *optimistic unchoking*. This slot is assigned via a 30 seconds round-robin shift over all the interested peers regardless of their upload rate. The protocol therefore creates a tit-for-tat data exchange based on the short-term behavior of a peer (i.e., the bandwidth it provides in return). Due to optimistic unchoking, new peers have a chance to obtain their first pieces of data and bootstrap the process.

3.2 Reputation-based unchoking

We use a more advanced unchoking policy which not only rewards short-term tit-for-tat behavior but also takes long-term reputation into account. The eventual effect of this should resemble the policy of `TVTorrents`-like central trackers where peers can only download if they have uploaded enough. In our reputation-based BitTorrent, peers only assign the upload slots to peers that have a reputation which is above a certain threshold δ. The threshold δ has a value below zero, so that new peers entering the system (who start with reputation zero at all other peers) and peers that upload as much as they download can participate normally. The more negative the value of the threshold, the more peers can download without uploading. When a peer keeps on downloading more than it uploads, its reputation at more and more peers will after a certain moment sink below the threshold value. From then on, downloads will take more and more time. A peer can easily prevent this from happening by seeding each file after it is downloaded until they have uploaded at least the size of the file to others. In this way, the average reputation of the peer will stay around zero, ensuring normal participation. If a peer wants to 'save' for the future, it can upload more than it downloads, creating a reputation far above zero.

3.3 Short-term versus long-term behavior

Reputation-based BitTorrent has a very efficient balance between the effects of short-term behavior and long-term reputation. On the one hand, the Tit-for-Tat performance of BitTorrent is guaranteed since peers that provide a low return bandwidth are still choked even if they have a high reputation. On the other hand, peers that have a low reputation but are improving their behavior by providing a high bandwidth can participate normally as soon as they have uploaded enough. Therefore, neither Tit-for-Tat nor reputation dominate the protocol.

4 Results

In this section we present the results of extensive trace-based simulations in which we analyze the effectiveness and accuracy of our reputation system, and compare our reputation-based BitTorrent policy is compared with traditional BitTorrent.

4.1 Simulation setup

We have built a trace-based simulator which incorporates all relevant aspects of BarterCast and BitTorrent. We simulate the epidemic peer discovery, the BarterCast statistics exchange protocol, and the reputation computation. Furthermore, we have implemented a BitTorrent simulator which simulates the protocol on piece-level including unchoking, optimistic unchoking, and rarest-first piece-picking. We have combined all processes in a trace-based simulation environment and have compared the performance and properties of different appraoches based on real network data. In our simulations, we use real network traces from the private BitTorrent tracker `filelist.org`, obtained in January 2006. The traces contain detailed behavior of all peers that were active in the file-sharing network, including uptimes, downtimes, connectability, and file-requests. For our simulations, we use traces of 100 peers active in 10 different swarms during one week. Since we do not have information about the real up- and download bandwidths of the peers, we currently simulate common ADSL users with a 3 MBps downlink and a 512 KBps uplink. We distinguish between *freeriders* that immediately leave the swarm after finishing a download and *sharers* that share every downloaded file for 10 hours. 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. The filesizes in the trace range from several tens of megabytes to about one to two gigabytes, representing mostly audio-files and movies.

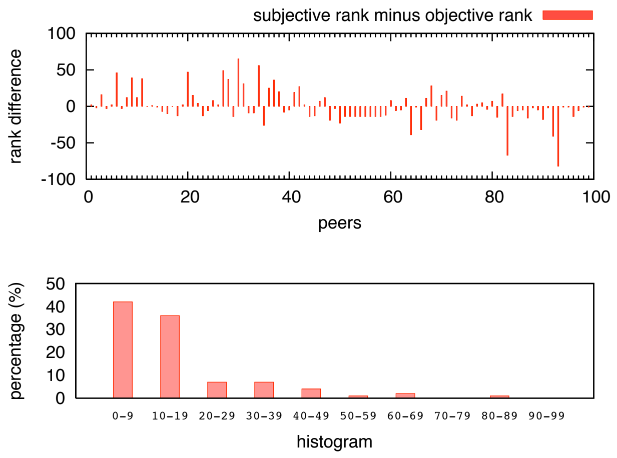

Figure 4: Accuracy of the reputation system; the difference between the objective and the subjective rank for all peers (top) and a histogram of this difference (bottom). In the top plot peers are sorted according to their objective rank.

4.2 BarterCast distributed reputation

As mentioned before, the reputation obtained by BarterCast is subjective due to its fully distributed nature. It is not impossible that a peer has a high reputation at one peer, while it has a low reputation at another. Given this subjectivity, our aim is that the reputation a peer *on average* has with other peers it encounters realisticly reflects its real, objective up- and downloading behavior. In our simulations, we do know the real behavior of peers as well as the subjective reputation that emerges in the system. We analyze the accuracy of the subjective reputation as follows: (1) we compute the average subjective reputation of all peers and rank the peers accordingly (*subjective rank*); (2) we compute the real (upload - download) for all peers and rank them accordingly (*objective rank*); (3) we compare the difference between the subjective and objective rank for each peer (*rank difference*). The smaller the rank difference, the more accurate the subjective reputation.

Figure 4 displays the rank difference for all peers, as well as a histogram. For 42% of the peers, the difference between the subjective and objective rank is less than 10%. Overall, a large majority of the peers (78%) has a difference less than 20%. In Figure 5, the average subjective rank of sharers and freeriders is compared for a system with 25% freeriders. When time progresses, the average rank of the freeriders clearly decreases, while the average rank of sharers increases. This is exactly the kind of effect we are aiming for with the reputation system. These results are quite spectacular, given that the subjective reputation is based on completely distributed information without any central component.

4.3 Reputation-based BitTorrent

We have performed extensive experiments to evaluate the our reputation-based BitTorrent policy. From Figure 5, discussed above, it is clear that the reputation of freeriders gradually decreases during the week. The effect of this decreasing reputation on the actual average download speed of the freeriders is displayed in Figure 6. The values in the plots denote the relative speed of the freeriders (i.e., the speed of the freeriders divided by the speed of the sharers). It is interesting to note that the speed of the freeriders at the first days is higher than that of the sharers. This is due to our earlier observations that freeriders have more upload bandwidth available for Tit-for-Tat, and therefore a higher download speed than the sharers. However, in our system this advantage of freeriders quickly vanishes when their reputation catches up and their speed becomes slower than that of the sharers. Finally, in Figure 7 the actual download speeds for various values of γ are displayed for both freeriders and sharers. The larger γ, the stronger the difference between the freeriders and sharers becomes. It can be observed as well that the download speed of the sharers gradually

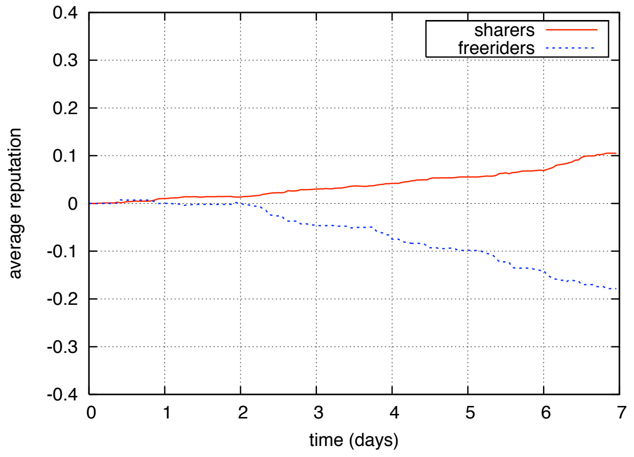

Figure 5: The average reputation over time for sharers and freeriders in a system with 25% freeriders.

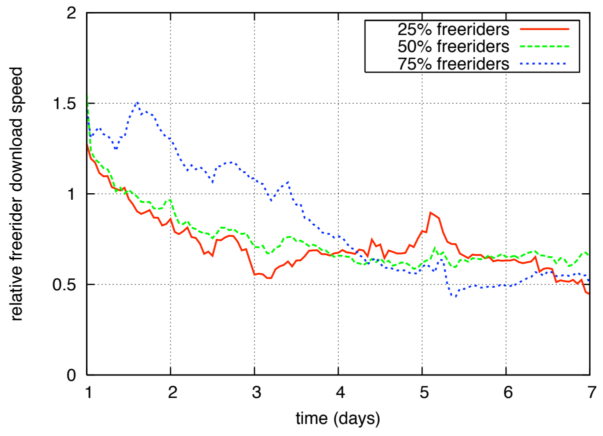

Figure 6: The relative speed of freeriders over time.

goes down when the policy becomes stricter. This is because when the freeriders are less successful in getting pieces their contribution in Tit-for-Tat also becomes less, which effects the sharers as well. On first sight, this seems as if the policy bites in its own tail. However, it is important to remember that the aim of sharing-ratio enforcement is to increase the performance by eventually having *more* sharers in the system. The 'punishment' of freeriders visible in Figure 7 will hopefully turn them into sharers, which increases the performance for every peer in the system.

5 Related work

A lot of mostly theoretical research exists into reputation, trust, and BitTorrent. First of all, in [?] the original tit-for-tat policy of BitTorrent is presented. In [?], a partially centralized reputation mechanism is presented which relies on authorized agents to keep track of the reputation. EigenTrust [?] is a well-known algorithm for globally consistent trust management, but is computationally heavy and not feasible for practical systems with high churn. In [?, ?, ?], token-based schemes are discussed that use centralized agents for administration of the tokens. In [?], an overview of important characteristics of large-scale modern day P2P networks is given, and the designs of several techniques are presented. However, the paper does not

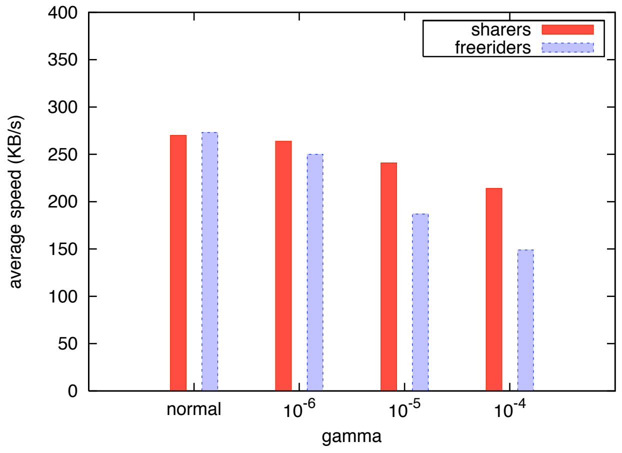

Figure 7: The average download speed of sharers and freeriders using normal BitTorrent and reputation-based BitTorrent with different values of γ and 25% freeriders.

discuss how to obtain the information in a distributed way, how to cope with newcomers, and how to apply these techniques to a specific peer-to-peer protocol like BitTorrent.

6 Conclusions and future work

We have presented a secure, fully distributed reputation system called BarterCast. Apart from a few unsuccessful initiatives this is the first deployed fully distributed reputation protocol that is both practically feasible and limits lying and collusion. We have presented a BitTorrent policy which makes use of these long-term reputation statistics by enforcing peers to contribute as much as what they download in the long term. We thereby overcome the problem of BitTorrent's lack of seeding incentives. The effect of this fully distributed sharing-ratio enforcement is comparable to that of central BitTorrent trackers like `TVTorrents`, where enforcement leads to large numbers of seeders and therefore a very high performance. In our system however, no centralized administration, authority, or technology is necessary. We have presented extensive simulations which show that the distributed subjective reputation successfully approximates the objective reputation of peers, that freeriders have a decreasing reputation in the system, and that reputation-based BitTorrent successfully 'punishes' freeriders as a motivation for them to become sharers. BarterCast has recently been deployed in the open source file sharing network Tribler [**?**, **?**]. Deployment results and further analysis based on real user data will be discussed in further work.