Mitigating the Effect of Free-Riders in BitTorrent using Trusted Agents

Alex Sherman, Angelos Stavrou, Jason Nieh and Cliff Stein
Department of Computer Science, Columbia University, New York, NY
asherman, angel, nieh, cliff@cs.columbia.edu

Abstract

Even though Peer-to-Peer (P2P) systems present a cost-effective and scalable solution to content distribution, most entertainment, media and software, content providers continue to rely on expensive, centralized solutions such as Content Delivery Networks. One of the main reasons is that the current P2P systems cannot guarantee reasonable performance as they depend on the willingness of users to contribute bandwidth. Moreover, even systems like BitTorrent, which employ a tit-for-tat protocol to encourage fair bandwidth exchange between users, are prone to free-riding (i.e. peers that do not upload). Our experiments on PlanetLab extend previous research [14, 12, 11] demonstrating that such selfish behavior can seriously degrade the performance of regular users in many more scenarios beyond simple free-riding: we observed an overhead of upto 430% for 80% of free-riding identities easily generated by a small set of selfish users.

To mitigate the effects of selfish users, we propose a new P2P architecture that classifies peers with the help of a small number of *trusted nodes* that we call Trusted Auditors (TAs). TAs participate in P2P download like regular clients and detect free-riding identities by observing their neighbors' behavior. Using TAs, we can separate compliant users into a separate service pool resulting in better performance. Furthermore, we show that TAs are more effective ensuring the performance of the system than a mere increase in bandwidth capacity: for 80% of free-riding identities a single-TA system has a 6% download time overhead while without the TA and three times the bandwidth capacity we measure a 100% overhead.

1 Introduction

Even though Peer-to-Peer (P2P) systems present a cost-effective and scalable solution to content distribution most companies that distribute videos, games and software online continue to rely on expensive infrastructures such as Content Delivery Networks (CDNs). The reason for this is that companies such as Yahoo and YouTube that depend on ad revenue must guarantee reliable download times to their users. However, the performance (*i.e. download time*) experienced by a P2P participant is heavily dependent on the willingness of other peers to share their bandwidth. Some P2P systems including BitTorrent[3], a popular P2P system, try to encourage pairs of neighbors to upload data to one another by using a "tit-for-tat" protocol. Unfortunately, both earlier research [14, 12, 11] and our experiments show that simple client modifications allow "free-riders" *(i.e. selfish nodes that have zero upload rate)* to complete their download. Furthermore, we show that when we have low and high contributing nodes in Bittorrent, there can be a significant increase in download times for the high contributing nodes.

This inherent Bittorrent weakness can lead to serious performance degradation from selfish behavior and expose BitTorrent to Sybil attacks [9]. In such attacks, a small set of users, or even a single user, participate in the system with multiple identities. The goal is to boost their download rate and thus decrease their download time. Such users can generate multiple identities in the system either by using multiple IP addresses or even connecting from a single machine through multiple paths via an anonymous network such as Tor [6]. Observe, that the software client that automates creation of many fake identities can be made widely available and adopted by many users. These fake client instances, which can potentially correspond to a very small set of real users, can seriously deteriorate the overall download times for the rest of the users rendering the system unusable for large files. In our PlanetLab experiments, we observed that in cases where free-riding clients make up 80% of the system the download speed of compliant clients can increase by 430%. This means that a compliant DSL user will spend an hour downloading a music video instead of 10 minutes. This degradation in performance is clearly unacceptable for companies that build their online businesses around video, software or games distributions and must guarantee reliable times in order to retain their users.

Our aim is to study Bittorrent's behavior in the presence of selfish clients and to propose architectural changes

that guarantee reliable performance for compliant users. To that end, we introduce a new class of peers called *Trusted Auditors* to the P2P system. Trusted Auditors (or TAs) participate in the system along with other peers and build reputation of their peers by observing their upload behavior. The reputation is tied to each client's identity which in turn determines the level of service received by this identity. This technique limits the damage from the free-riding identities. Thus, we guarantees a high level of service for compliant users. We rely on a set of TAs managed by the content provider to establish reputation rather than a peer-based feedback approach in order to avoid collusion by a large group of peers that may occur when all malicious clients are generated by a single user. We also show that this technique of using *Trusted Auditors* is more effective than simply provisioning more server bandwidth in the P2P system. We now give a brief overview of BitTorrent before explaining our additions to the system.

BitTorrent Overview A file that is distributed over a BitTorrent system is broken up into a number of smaller chunks (typically 256KB in size). BitTorrent leverages the P2P users' upload bandwidth by having them upload these chunks to one another. A BitTorrent system typically contains one or more *Seeds* or clients that contain all the chunks of the file. A BitTorrent system also includes a *Tracker*. Clients joining the system contact the tracker to obtain IP addresses of other clients. By default the tracker hands out upto 50 IPs selected at random. A Bit-Torrent client implements a tit-for-tat heuristic to foster fair data exchange between clients. Based on the official tit-for-tat specification [2] a BitTorrent clients *unchokes* (or allows uploads to) four peers, from which it receives fastest download rate. The client updates this set of peers every 30 seconds and also selects an additional peer to unchoke at random. This latter part intended to help the client with discovery of faster peers is known as *optimistic unchoking*. A full description of the BitTorrent behavior can be found in [2].

Our Approach In the presence of many free-riding clients the performance of compliant peers suffers tremendously in BitTorrent for two reasons. First, since the mapping of clients and Seeds to one another is random, free-riders consume much of the Seeds' upload bandwidth and compliant peers receive a smaller inflow of new file chunks from the Seeds. Second, *optimistic unchoking* algorithm is very slow [11] at helping compliant peers to discover one another.

Our proposed design addresses these problems by identifying free-riders and separating free-riders and compliant users into different service pools. Inside their service pool compliant users get the full benefit of the Seed upload bandwidth and also have easier time discovering fast peers via *optimistic unchoking*. In order to identify free-riders we introduce a new class of peers called *Trusted Auditors* (or simply TAs). TAs are a set of peers managed by the content provider. They download the file just like regular clients, but in addition they gather information about bandwidth contribution rates of their neighbors (i.e. peers assigned to them by the tracker). In this way TAs help the system build a reputation history of each peer based on their bandwidth contribution rate. In addition, the system enforces strong client identities via password-protection. Client identities with good reputation are assigned to the same service pool isolated from the free-riding clients that have poor reputation.

We now explain why we believe our approach to be the right solution to the free-rider problem by addressing several alternatives and counter-arguments:

If the Seed bandwidth becomes the scarce resource in the case of many free-riders why not resolve it simply by provisioning more Seed servers?. It turns out based on our experiments that a system with fewer Seeds and TAs is more resilient against attacks by free-riders. For example, in our experiments on a 100-client BitTorrent system we found that even after provisioning 20 times the required seed bandwidth enough free-riding identities can cause a 100% increase in the download times of compliant users. At the same time we show that a system of one Seed and one TA are sufficient to identify and remove a great majority of free-riders in the system of the same size. Thus the total amount of bandwidth that the content provider needs to provision to implement TAs is significantly less and more effective than the extra Seed servers.

Why need an extra set of TAs and not establish reputation based on feedback from other Peers?. Peer-based reputations systems [7, 10] do not address the problem of collusion by a large set of free-riding peers who can collude to inflate one another's reputation. A malicious user who creates a majority of free-riding identities can easily have them collude.

Why is it sufficient to only separate the free-riders and not split the rest of the users based on their specific contribution level ? Once we filter out free-riders (and very low-contributing peers) we can use the results of Legout et.al [1]. Although their work does not study the free-riders they show for other scenarios peers with similar upload rates tend to cluster together in BitTorrent. We believe that we can use our methods to improve the results in [1] further by creating multiple service pools, but we leave that as an item for future work.

Finally, what if the malicious peers can learn the IPs of TAs and send data just to those IPs in order to raise their reputation? We suggest two reasons why it would be difficult to do that. First, the behavior of TAs is indistinguishable from other clients. Outwardly they

implement the exact BitTorrent protocol. Second, the system of a large content provider employs many TAs that frequently renew their IP addresses using dynamic DNS. The study of the reliability of these services is beyond the scope of this paper.

Contributions Our contributions are:

- We present results from our PlanetLab experiments that study the effect of a large number of free-riding identities on compliant users including the case of many Seeds.
- We present a design sketch of the system that uses Trusted Auditors to guarantee fast download times to good peers in the face of free-riders.
- We present measurement results that demonstrate the effectiveness of TAs in identifying malicious peers.

We present the design of our system in section 2. Our measurement methodology is described in section 3. Experimental results are presented in section 4. We review related work in section 5. We conclude and list future work in section 6.

2 System Design

In this paper we sketch out the design of our system and carry out the measurements on the effectiveness of Trusted Auditors. We leave the implementation of the rest of the system components to future work. In brief, we augment the existing BitTorrent design by adding a set of TAs that help identify free-riders and low-riders (i.e. peers contributing very little bandwidth). The compliant clients and free-riders are separated into distinct *service pools*.

In future work we plan to take this idea further and differentiate compliant clients into multiple service pools based on their specific upload rates. However, we find that the biggest gain of our system comes from filtering out free-riders and low-riders because they significantly constrain data flow to compliant users. For this reason our current design and measurements focus on two classes of users: selfish and compliant. We now discuss the service pools, TA functionality and implementation of strong client identities.

Service Pools In its most basic state for each downloadable file our system contains three service pools: *free-rider pool* for free-riders, *compliant pool* for compliant clients and *fresh pool* for newly joining identities who do not yet have a sufficient reputation history in the system. Each service pool contains its own Seed servers, a tracker and a set of TAs.

Evaluation by TAs A small percentage of the clients in each pool are the Trusted Auditors. TAs evaluate their neighbors based on the download rate from those neighbors weighted by the amount of time that the TAs themselves *unchoke* (or push data to) that neighbor. TAs submit the clients' reputation statistics to the local *tracker* that maintains each client's history. Each *ta_epoch* the tracker examines the clients' history and may decide to move a client between service pools. The number of TAs (determined by the *percent_ta* parameter) and the time they take to make a decision to move a client (determined by *ta_epoch*) present a tradeoff between how quick vs. how accurate is the classification. In our measurements we found that in a 50 client system with 1 TA (*percent_ta* = 2) over 20 minutes (*ta_epoch* = 20) we can reliably identify 75-80% of the free-riders. This means that say for a large movie download that may take 2 hours even with a large number of free-riding identities, compliant clients that join the system for the first time only need to spend a relatively short time in the fresh pool before being placed in the compliant pool. Returning compliant clients are placed immediately in the compliant pool.

Observe, that the classification of the clients need not be fully accurate. As long as the TAs can identify and separate a large fraction of free-riders they can significantly improve the performance of the remaining compliant users. Further to make up for classification errors TAs continue to evaluate all the clients in the free-rider and compliant service pools and make corrective moves.

The history of clients' contribution is weighted by time over multiple downloads by that client. However, the history is heavily weighted towards the most recent *ta_epoch*. Although we do not report these measurements here due to lack of space we observe that such weighing of the history makes the system very effective against *smart* free-riders who may for instance pretend to be compliant in the fresh pool and then turn the free-rider behavior on. We leave the study of such strategies to the future work.

Strong Client Identities A client's reputation is tied to her identity. To maintain strong identities after the client selects to download a file they log in to the system with a username and a password so that the history associated with the identity can be retrieved from the system. Upon login the client receives a signed credential and a verification key. We modify the protocols to have peers verify one another's credentials. When a compliant peer A receives a connection from peer B it is in the interest of peer A to adhere to the protocol and to verify that B is authorized by the system to be in its compliant pool.

3 Methodology

We performed two sets of measurements on the PlanetLab. The first set was to evaluate the behavior of the standard BitTorrent system with free-riders. The second set was to evaluate the effectiveness of TA(s) in identifying the free-riding clients. In all our experiments we used 50 BitTorrent clients that were downloading a 64MB file. For different experiments we would add different number of Seeds (1,2,3,5,10) and configured a different percentage of the 50 clients to act as free-riders (0,20,40,60,80%). The Seed service were initialized with the 64MB file and the clients would begin the download simultaneously. The 64MB corresponds to a typical size of a music video file.

To implement free-riders we modified the BitTorrent clients not to send not to announce or upload data chunks. To make sure that our results were not skewed by bad clients we carefully selected PlanetLab machines that were capable of uploading at this rate.

We performed three experiments for each set of parameters and measured the download completion times of the compliant users. In each test, we randomly configured three of the compliant users to act as Trusted Auditors. The download/upload behavior of a TA did not differ from that of a compliant user and thus having TAs did not skew any download measurements. Each minute TAs logged the bytes that they sent and received from each of their neighbors. By analyzing the TAs' logs after each experiment we could then evaluate how precisely and how quickly could a subset of 1, 2 or all 3 of the TAs identify the free-riders.

4 Evaluation

4.1 Free-rider Problem

In the first part of the evaluation, we performed experiments to assess the impact of selfish participants with zero or low upload rate, on the performance of compliant bittorrent clients. Table 1 presents the increase in the average download time as we vary the percentage of free-riders in a 50-client system for a 64MB file download. The upload bandwidth capacity of the seed and the compliant clients was limited to a maximum of 25KB/s. We observe that even for a mere 20% of free riding identities, we have a 25% time overhead. In practice, 20% of free-riding identities correspond to 10 fake identities for a system of 50 nodes and can be easily generated by a single user. To make things worse, when 60% and 80% of the identities behave selfishly as free-riders, the average download times increase by 146% and 430% respectively. The more free-riding identities there are in the system the more of the Seed upload bandwidth they use without sharing it with the rest of the system. The secondary effect is that compliant user has a hard time distinguishing between good and bad users since the compliant users have little new data to share.

Our experiments show that the effect of selfish users on compliant users is extend beyond the case of free-riders. Table 2 shows the effect of low-riders (contributing 3KB/sec) on compliant users and a Seed that contribute at 80KB/sec. Here for 40% and 80% of low-riders the slowdown for the compliant users is 18% and 47% respectively. A 47% slowdown corresponds approximately to an increase of a movie download from 3 hours to 4 and a half.

% Free-Riders	Mean Download	% Increase
0%	2690	0%
20%	3370	25%
40%	4440	65%
60%	6620	146%
80%	14270	430%

Table 1: Percentage of increase in mean download time (seconds) with free-riders in a 50-client system with 1 Seed

4.2 Trusted Auditors

In this section, we evaluate the effectiveness of TAs in classifying the free-riders in the system. TAs attempt to identify as many free-riders as possible with as few as possible false positives (i.e. compliant users that do not push data to these TAs at a reasonable rate). To reduce false positives to negligible levels (below 0.01%), we consider only those clients whom TA(s) unchoke (i.e. push data to) for a considerable period of time. Our experiments indicate that having TAs unchoke a node for a combined time of one minute without that node's reciprocation with a reasonable upload rate is sufficient to classify it as a free or low rider (Observe that BitTorrent uses 30 seconds as a conservative estimate during its optimistic unchoking).

% Low-Riders	Mean Download	% Increase
0%	874	0%
20%	933	7%
40%	1034	18%
60%	1128	29%
80%	1282	47%

Table 2: Percentage of increase in mean download time with increase in low-riders in a 50-client system with 1 Seed. Low-riders upload rate: 3KB/s. Other clients: 80KB/s

Figure 1 depicts the percentage of free-riders (y-axis) correctly identified by TAs over time (x-axis) in a 50 client system with 80% free-riders. Three different curves correspond to different number of TAs used. Although 3 TAs identify free-riders faster even 1 TA can identify 75% of those free-riders after 20 minutes even with 80% of free-riders in the system. This means that after 20 minutes

75% of the free-riders can be identified and filtered out and performance for the compliant users can be immediately improved. (We found similar results for identification of low-rides but we do not report them here due to space limitations).

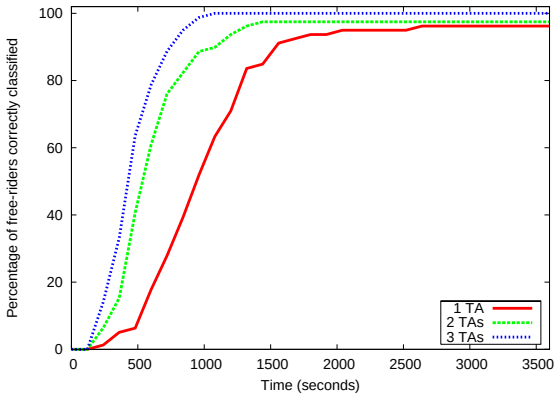

Figure 1: Percentage of free-riders correctly identified over time with a system of 50 clients, 80% of which are free-riders. Different curves correspond to the different number of TAs used

Based on the TA identification rate figure 2 compares download times of our system with 1 TA vs BitTorrent provisioned with multiple Seeds. Although the TA-based system takes a slight hit in the beginning to identify the free-riders it beats the setup with 10 Seeds starting with 200MB file size. Thus with a fifth of the provisioned resources (1 Seed + 1 TA vs. 10 Seeds) our system is expected to perform better for 200MB+ sizes. It behaves better than a 3 Seed system even for smaller file sizes. We also observe that the returning users who already have a reputation history in the system immediately see the improved download times.

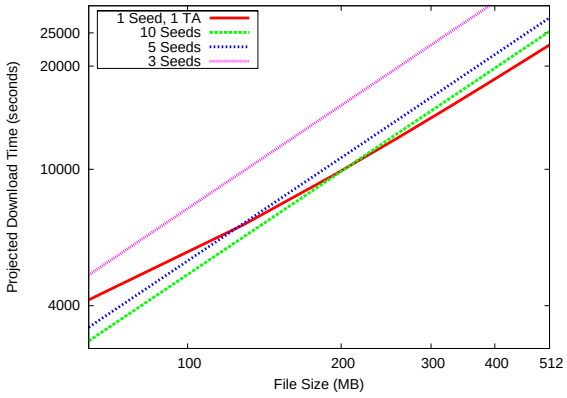

Figure 2: Mean download times for various file sizes in a 50-client system with 80% free-riders. Compares our system (1 Seed, 1 TA) vs multiple Seed provisioning.

4.3 Multiple Seeds

Figure 3 shows the effect of free-riders on the system with various number of seeds and TAs. For each percentage of free-riders (x-axis) the graph shows the overhead factor (y-axis) on the mean download time when these free-riders are present. The system with just one seed and 1 TA is much more resilient to free-riders than any other setup including one with 10 seeds. In fact while other setups have at least 10, 20 and 40% overhead for 40,60 and 80% free-riders respectively the system with a TA has under 6% overhead. The only overhead for the case with 1 Seed + 1 TA is the time taken to identify free-riders which gets amortized over a large file download.

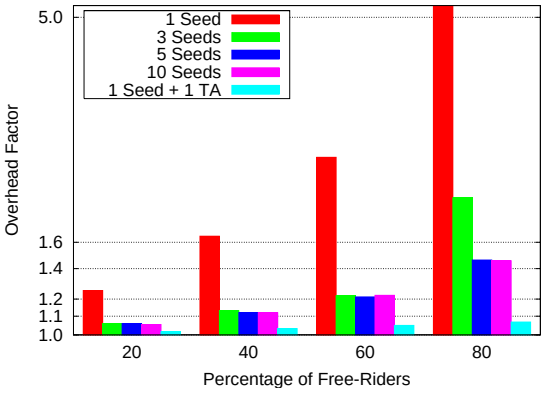

Figure 3: Effect of free-riders in systems with multiple seeds as well as the system with 1 Seed + 1 TA for a 512 MB file download. The y-axis shows the overhead factor of the mean download time with a given percentage of free-riders vs. the same setup (Seeds, TAs) without free-riders

Finally, we comment on the distribution of the download times in the case of free-riders with multiple seeds (figure 4). Without any free-riders, we observed that all clients complete almost simultaneously while with free-riders the tail of the distribution takes an especially heavy hit. While the mean goes up by about 40% the 90th percentile takes almost double the time to finish the download. This means that even in a system that is well provisioned with Seed bandwidth the content provider can expect that at least some users will see a doubling of their download times with free-riders.

5 Related Work

Free-Riding in Bit-Torrent Qiu and Srikant [4] point out in their analysis that the *optimistic unchoking* behavior in BitTorrent [2] opens up the client to losing upto 20% of its bandwidth to free-riders who simply need to wait for other peers to "optimistically" connect to them.

Recent research [14, 12, 11], has demonstrated how in practice a modified BitTorrent client can exploit different strategies to achieve better performance. In [14]

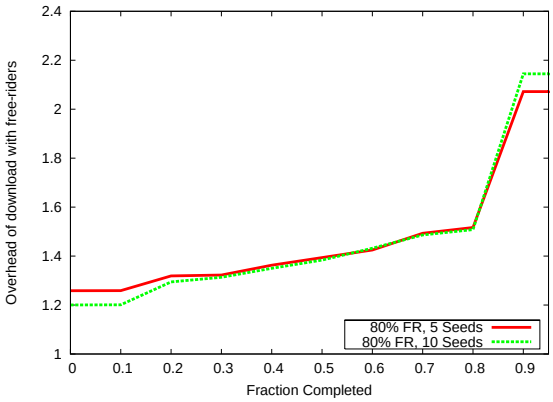

Figure 4: Overhead in completion times of compliant peers with a system of 50 clients + multiple Seeds, and 80% of free-riders as compared to mean download time without free-riders

authors implement a BitThief client that, similar to our free-riding client, does not contribute data to others. They demonstrated that BitThief can perform better than regular clients with scenarios where majority of the client run the regular BitTorrent. *BitTyrant* client [11] takes advantage of some of the altruism of other BitTorrent clients to send minimum possible data and achieves a 70% performance improvement. LargeView Exploit [12] implements a free-riding client that connects to as many IPs as possible. This is perhaps the closest to our work. [12] shows that their free-riding clients can perform on par with the rest of the clients even where free-riders are upto 40% of the network. We focus our work on the study of the degradation of service for compliant users in the face of free-riders. We also extend the study to show that adding significant Seed bandwidth does not negate the effect of a large percentage of free-riders.

Legout et.al [1] study the clustering behavior of BitTorrent and demonstrate that peers that upload at the same rate tend to cluster with one another. We were able to confirm the results of their scenario where the lowest class of peers still upload at 5KBytes/s. However, they do not consider the case of free-riders or peers that upload at a very low rate (1-2KBytes/sec).Our experiments show that in those cases the flow of new data can be significantly constrained and clustering does not play a role.

Reputation and Incentives Some systems such as [13] use virtual credits to incentivize peers in P2P systems to exchange services fairly. Both ARA and Dandelion provide secure methods for credit-checks and peer validation. [5, 8] go a step further and implement actual currency payment for rendered services among peers. However, with a large number of malicious users who do not share data the compliant users will need to waste a lot of resources and overhead of credit checks only to discover that their neighbors are cheating. Reputation systems such as [7] and PeerTrust [10] rely on feedback from other peers to establish trusted peers who provide good service. Such systems, however, are open to collusion especially where majority of the peers are malicious and can to hype one another reputation. In contrast, our system avoid collusion by utilizing TAs, a managed set of peers.

6 Conclusion and Future Work

We presented a study of the effect of selfish users on compliant users in BitTorrent. Through extensive experimentation on Planet-Lab, we demonstrate how such selfish behavior, which can span from zero to low content contribution, can deteriorate the overall system performance by causing a 4-5 times increase in compliant users' download time. To address the effects of selfish behavior, we presented a novel architecture that employs Trusted Auditors to identifying and separate compliant from selfish clients. Our preliminary results demonstrate that our approach can ensure the download times of regular clients even when we have 80% of free-riding identities in the system.

As a future work, we plan to extend our measurements and present a full implementation of our system that includes identification and separation users into multiple download classes. Furthermore, we would like to study the overall effect of such a separation on user performance and to the overall system. Finally, we want to consider scenarios where the selfish users attempt to game the system using dynamic behavior by initially contributing bandwidth to obtain admittance to a good service pool.

References

[1] A.Legout, N.Liogkas, E.Kohler, and L.Zhnag. Clustering and sharing incentives in bittorrent systems. In *SIGMETRICS*, 2007.

[2] B.Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of P2P Systems*, 2003.

[3] Bittorrent. `http://www.bittorrent.com`.

[4] R. D. Qiu. Modeling and performance analysis of bittorrent-like peet-to-peer networks. In *SIGCOMM*, 2004.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Reputation In P2P Anonymity Systems. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[6] R. Dingledine, N. Mathewson, and P. Syverson. TOR: The second generation onion router. In *Usenix Security*, 2004.

[7] F.Cornelli, E.Damiani, S. C. di Vimercati, S.Paraboschi, and P.Samarati. Choosing reputable servents in a p2p network. In *WWW*, 2002.

[8] D. R. Figueiredo, J. K. Shapiro, and D. Towsley. Payment-based incentives for anonymous peer-to-peer systems. Technical report, July 2004.

[9] J.R.Douceur. The sybil attack. In *IPTPS*, 2002.

[10] L. L.Xiong. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. In *IEEE Transactions on Knowledge and Data Engineering*, July 2004.

[11] M.Pieatek, T.Isdal, T.Anderson, A.Krishnamurthy, and A.Venkataramani. Do incentives build robustness in bittorrent. In *NSDI*, 2007.

[12] M.Sirivianos, J.H.Park, R.Chen, and X.Yang. Free riding in bittorrent networks with the large view exploit. In *IPTPS*, 2007.

[13] S. M.Sirivianos, X.Yang. Dandelion: Cooperative content distribution with robust incentives. In *USENIX*, 2007.

[14] T.Locher, P.Moor, S.Schmid, and R.Wattenhofer. Free riding in bittorrent is cheap. In *5th Workshop on Hot Topics in Networks*, 2006.