Free-riding in BitTorrent Networks with the Large View Exploit

Michael Sirivianos Jong Han Park Rex Chen Xiaowei Yang
Department of Computer Science
University of California, Irvine
{msirivia,jonghanp,rex,xwy}@ics.uci.edu

***Abstract*— This paper presents an experimental study on the behavior of BitTorrent networks when selfish peers attempt to maintain high download rates without uploading. We modified a BitTorrent client so that it acquires a larger than normal view of a BitTorrent swarm and connects to all peers in its view. At the same time, the modified client does not upload any data to its peers. Our experimental results show that: a) our modified free-rider client can achieve better download rates than a compliant client in most common-case public torrents; b) when the percentage of our modified free-rider clients in PlanetLab-residing torrents with ∼300 leechers is less than 40%, free-riders on average outperform compliant clients; and c) as the number of free-riders increases, both free-riders and compliant clients incur substantial performance degradation. These results suggest that the large view exploit is effective, and it has the potential for wide adoption.**

I. Introduction

Peer-to-peer (P2P) content distribution networks are powerful systems that utilize the bandwidth resources of their users. The performance of a P2P network is highly dependent on the users' willingness to contribute their bandwidth. However, selfish (rational) users tend not to share their bandwidth without incentives. A study that revisited the free-riding issue in Gnutella revealed that 85% of the peers do not share any files [10].

The popular BitTorrent P2P protocol [7] employs the rate-based tit-for-tat incentive mechanism to motivate users to upload. In theory, the data exchange between two BitTorrent peers can be modeled as an iterated prisoner dilemma game, and tit-for-tat is shown to be the winning strategy that optimizes a player's payoff [5]. However, it has been suggested that BitTorrent's incentives are vulnerable to manipulation [11, 13, 16], because it is difficult to enforce strict tit-for-tat in practice.

We contribute to the study of BitTorrent manipulation by revealing a new aspect of the free-riding problem. We show that clients can refrain from uploading any data and still achieve better downloading rates than tit-for-tat compliant clients. If a selfish client attains a much lower downloading rate when it free-rides than when it uploads, the selfish client has less incentive to free-ride. On the other hand, if a free-riding client can achieve a better or only slightly worse downloading rate than a tit-for-tat compliant client, a selfish client may have incentive to free ride. In practice, users may value their uplink bandwidth more than slightly faster download rates, thus they may be motivated to free-ride. For example, clients with access providers that impose quotas on outgoing traffic or clients with limited uplink bandwidth (e.g. 1.5Mbps/128Kbps ADSL) may wish to save their uplink for other critical tasks. A dire prediction is that if more and more users start to free-ride, BitTorrent communities will experience the "tragedy of the commons," manifested as system-wide performance degradation in BitTorrent networks.

We design, implement and experimentally evaluate a new BitTorrent free-riding technique, the large view exploit: a selfish BitTorrent client acquires a larger than normal view of the torrent (swarm), and connects to all peers in its view. In this way, the client increases the likelihood of becoming unchoked by leechers and discovering seeders. At the same time, the selfish client never uploads any data to its peers.

We run experiments with modified BitTorrent clients using this exploit in public torrents and PlanetLab-residing torrents with ∼300 leechers. Our results show that in most public torrents, a free-riding client performs better than its tit-for-tat compliant counterpart. In PlanetLab-residing torrents, when the torrent consists mainly of compliant clients (>60%), free-riders on average outperform compliant clients. Moreover, as the number of free-riders increases, both compliant and free-riding clients experience substantially increased downloading times. These results suggest that the large view exploit has the potential to be widely adopted; if the client population is not dominated by free-riders, the download rates of free-riders are high when compared to those of compliant clients.

The rest of this paper is organized as follows. Section II compares our work with related work. Section III describes the design rationale and implementation of the large view exploit. Section IV presents experimental results. We discuss a technique to prevent the exploit in Section V and conclude in Section VI.

II. Related Work

Several studies have conjectured and experimentally demonstrated the free-riding phenomenon in BitTorrent. Jun et al. [11] show that in a PlanetLab-residing swarm with ∼150 leechers, 75 free-riders can attain average download completion times, almost as good as tit-for-tat compliant peers. However, they define a free-rider as a peer that uploads with rates up to 4KB/sec, while other peers upload with rates up to 100KB/sec. In contrast, our work shows that a client using the large-view exploit can download as fast as or faster than a compliant client without uploading any data.

Liogkas et al. [13] show that peers that discover many seeders in a swarm can connect to seeders only, thus substantially increasing their download rates without contributing bandwidth. In addition, Andrade et al. [4] demonstrate that free-riders may have better sharing-ratio (the amount of uploaded data divided by downloaded data) than compliant peers when a swarm has many seeders.

Schneidman et al. [16] conjectured that a peer can increase the frequency with which it gets optimistically unchoked by presenting multiple identities to a tracker. As a result, the tracker reports this peer to other peers multiple times, and other peers are now more likely to unchoke this peer. In addition, they state that a peer can probe other peers to more frequently optimistically unchoke him by reconnecting to them and getting better placement in the unchoking queue. The large-view exploit we implement does not require a peer to present multiple identities.

The weakness of BitTorrent's incentives presented in this paper has been experimentally demonstrated in two very recent works that were almost concurrent with ours. Locher et al. [14] describe and evaluate an exploit that bears much similarity to the large-view exploit described in this paper. Piatek et al. [15] showed that a client that connects to many peers, and carefully selects its peers and per-peer upload rates can achieve significantly higher download rates than mainstream clients, while substantially reducing its uplink bandwidth utilization.

Our paper complements Locher et al.'s work and offers additional insights in the following ways (we list the most important ones). First, we test our exploit in medium-scale private torrents on PlanetLab (∼300 leechers). These experiments allow us to systematically study how the number of free-riders affects system-wide performance in BitTorrent swarms. Locher et al.'s work presents an experiment for a small private torrent consisting of 4 leechers and one seeder. Second, we modify an existing BitTorrent implementation only with respect to the aspects of our exploit. In contrast, the free-rider "BitThief" in [14] is a client built from scratch. As a result, it inherently has features that are different from mainstream clients (e.g. it always uses random chunk selection instead of rarest first). We expect that our approach further demonstrates the effects of the large view exploit in isolation from other aspects of BitTorrent's implementation. Third, our work investigates the effectiveness of disconnecting from and reconnecting to leechers after having been unchoked by them, while Locher et al.'s work investigates whether uploading garbage to peers and sending false announcements of available chunks to them can yield performance gains. Fourth, we provide a more in-depth explanation of our exploit in the game-theoretic framework of the tit-for-tat strategy. Last, since both studies derive conclusions that concern a widely used content distribution system, our work's further validation of their insights is of particular importance.

III. The Large View Exploit

This section describes the design rational of the large view exploit and its implementation. For ease of exposition, we first summarize how Bittorrent works.

A. How BitTorrent Works

The BitTorrent protocol involves three parties: the server of the torrent file, the tracker, and the client. The torrent file contains meta-data information of the file to be downloaded, which includes the tracker's URL, the file's name and length, and the SHA-1 hash values of individual file chunks. A *tracker* maintains a list of all the clients that are currently downloading a certain file (*leechers*) or have the complete file and only upload it to others (*seeders*). The tracker, the leechers, and the seeders constitute a BitTorrent *swarm* (also referred to as *torrent*). To download a file, a client: 1) obtains the corresponding torrent file; 2) contacts the tracker to obtain a partial swarm view, which usually consists of up to 50 peers; 3) connects to the peers in the partial view; and 4) downloads file chunks from the seeders and/or exchanges file chunks with the leechers.

B. Design Rational

BitTorrent clients use a tit-for-tat scheme for chunk exchanges: a client always cooperates in the first move by uploading to another peer (*optimistic unchoking*). Thereafter, it uploads to peers that reciprocally upload to it. Cohen [7] describes that this strategy leads to cooperation, as the data exchange between two peers can be modeled as a repeated prisoner dilemma game and tit-for-tat is the winning strategy [5].

However, in BitTorrent, a client plays a finite number of rounds of the iterated prisoner's dilemma game with each of its peers. It can abandon the game once its peers have cooperated a certain number of times, i.e. when it has downloaded from its peers all the content that it needs. Consequently, if the client can play the game with many players from a large population of tit-for-tat compliant players, the best strategy may be to not cooperate and abandon the game with each peer after the first round. By not cooperating in the first move, it exploits the initial offers of the tit-for-tat compliant players and obtains a large portion of the needed content without incurring any cost. In addition, BitTorrent's implementation does not strictly abide by the tit-for-tat strategy due to the tradeoff between performance and susceptibility to free-riding.

At any time, a BitTorrent leecher unchokes (uploads to) n clients (typically four to ten). Among those n peers, the $n-1$ are the peers that are the fastest uploaders and are also interested in the leecher's content. The leecher revises its list of unchoked peers every 10 seconds and optimistically unchokes *one* peer every 30 seconds. 10 and 20 seconds after the last optimistic unchoking, the leecher samples the upload rates of all its interested peers, except of the one that it optimistically unchoked. If a sampled and currently unchoked peer is among the fastest $n-1$ uploaders, the leecher keeps that peer unchoked. Otherwise, the leecher chokes that peer and unchokes another sampled peer that is now among the $n-1$ fastest uploaders. 30 seconds after the last optimistic unchoking, the leecher samples the upload rates of *all* its interested peers and keeps unchoked only the $n-1$ fastest uploaders among them. It also optimistically unchokes one previously choked peer regardless of that peer's upload rate. Optimistic unchoking allows a leecher to discover peers that possess content of interest and that may be able to upload to it at higher rates than the currently unchoked peers.

BitTorrent seeders favor for unchoking the fastest downloaders or the most recently unchoked peers, regardless of whether the downloaders are cooperative with other leechers.

Based on the above observations and previous work on BitTorrent exploitation [11, 13, 16], we conclude that even if a client does not upload to its peers, it may be able to download at rates equal to or *higher* than those of tit-for-tat compliant clients. In a sufficiently large swarm, a client that connects to many more peers than the protocol specifies can increase the likelihood of becoming optimistically unchoked, as more peers have it in their list of candidates to unchoke. It can also find more seeders, which do not abide by the tit-for-tat rule.

C. Implementation

Drawing from the above conclusion, we implement a BitTorrent client that employs a new free-riding technique, the *large view* exploit, as follows:

1. It *never uploads* any file chunks to its peers.

2. It initiates a connection to the tracker every 15 seconds to mimic the behavior of a new client joining the swarm, and repeatedly requests and obtains partial swarm views.

3. Connects to all peers in its larger than normal swarm view.

The tracker could maintain state for each client and use authentication mechanisms to limit the rate with which clients obtain partial swarm views in step 2. However, modified clients could assume multiple identities (Sybil attack [8]). In addition, clients can exchange peer lists in order to widen their view of the swarm. We note that the latter mechanism is already incorporated and validated in mainstream and benevolent BitTorrent implementations [3] to improve the resilience of the system in the event of tracker failure.

Certain BitTorrent implementations, such as CTorrent [2], select peers to become unchoked according to criteria that allow new clients to quickly become uploaders [17]. A CTorrent client does not optimistically unchoke peers that have been recently optimistically unchoked. In contrast, it favors for optimistic unchoking peers that: a) claim to have no pieces; b) claim to have pieces that the client is missing; c) have been choked for the longest period of time; and d) in the past, have uploaded the most content to the client. Therefore, we consider an additional step that a free-rider may employ:

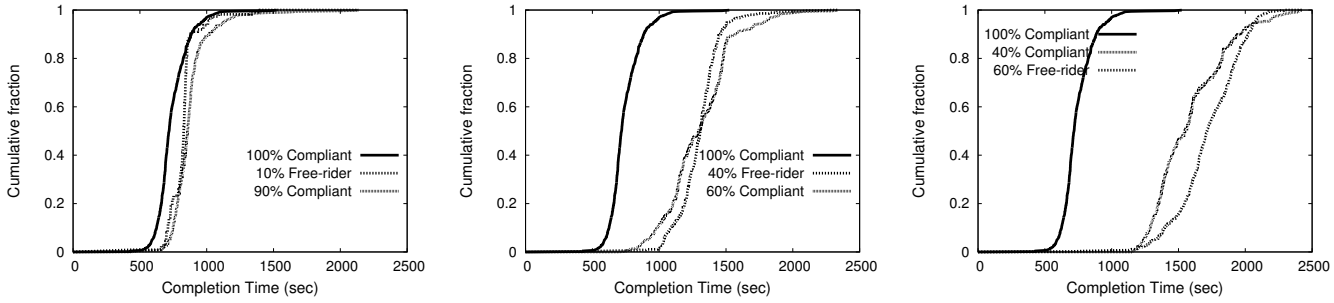
4. Disconnects from leechers and reconnects to them after the leechers have unchoked it, uploaded data to it, and then choked it again. This causes the leechers to remove any reference to the client's past transactions with them.

Step 4 has the potential to yield additional gains, as it may further increase the frequency with which a free-riding client becomes optimistically unchoked. We call this step *whitewashing* [9] with leechers.

IV. Evaluation

In this section, we evaluate the effectiveness of the large view exploit and demonstrate its detrimental impact on the system-wide performance of BitTorrent swarms.

We use BNBT[1] and Enhanced CTorrent 1.3.4 [2] as the BitTorrent tracker and client, respectively. Unless noted otherwise, in our experiments, free-riders are implemented as described in Section III-C, except that they do not employ whitewashing with leechers (they do



(a) (b) (c)

Fig. 1. **The BitTorrent large view exploit in PlanetLab-residing torrents for a 12MB file. Free-riders acquire a large view and connect to all peers, *without* performing whitewashing with leechers. There is one initial seeder and ∼300 leechers. All leechers' are rate-limited to download and upload at 30KB/sec: (a) 10% free-riders; (b) 40% free-riders; and (c) 60% free-riders. The mean download completion times over all clients of each type are: (a) free-rider 823 sec and compliant 872 sec; (b) free-rider 1299 sec and compliant 1290 sec; and (c) free-rider 1720 sec and compliant 1581 sec. All figures include a plot for swarms with no free-riders for the purpose of comparison. The mean download completion time in the swarms without free-riders is 741 sec.**

not perform step 4). We perform experiments both for PlanetLab-residing torrents and for public torrents. For the former, we spawn VServers on multiple PlanetLab [6] nodes. For the latter, one compliant and one free-rider client run on two distinct machines, which have exactly the same configuration and reside in the same LAN. Compliant clients connect to at most 50 peers. Recall that free-riders do not upload any data.

A. PlanetLab-residing Torrents

Figure 1 illustrates the impact of the large view exploit in PlanetLab swarms. In each experiment, there is *one initial seeder* and a mixed population of ∼300 compliant and modified leechers, running on distinct PlanetLab nodes. The leechers and the initial seeder join the swarm almost simultaneously and upon download completion, compliant leechers remain online to seed the file. In all cases, the compliant and the free-riding leechers download a 12MB file. We rate-limit all clients at 30KB/s for both upload and download to ensure that they have equal resources. Free-riders obtain a swarm view of ∼250 clients on average after ∼150 seconds in the downloading process. For every experiment configuration, we strive to involve the same PlanetLab nodes, and derive mean file download completion times for each node. For every node, we collect 8 to 10 measurements, depending on the node's availability.

As we observe in Figure 1, the download completion times of clients increases substantially as the percentage of free-riders increases. In Figure 1(a), when there are 10% free-riders, free-riders have shorter download completion times than compliant clients. For 40% free-riders, Figure 1(b), we observe that free-riders have download times that are on average almost equal to the ones of compliant clients. We also observe that the standard deviation of downloading times for compliant peers is larger than the one of free-riders. We believe that this is a consequence of the fact that compliant clients connect to different subsets of peers in the swarm, whereas large view free-riders connect to almost the same set of peers, which approximates the entire swarm.

When 60% of the client population are free-riders, Figure 1(c), the compliant clients outperform the free-riders. This is because despite their large views, free-riders have difficulty in making the fewer compliant peers to upload to them, as the probability of getting optimistically unchoked is substantially reduced. On the contrary, compliant peers upload, thereby prompting other compliant peers to unchoke them. Eventually, compliant clients form clusters in which they download faster by employing tit-for-tat.

B. Public Torrents

Figure 2 shows the downloading time of our free-riding CTorrent client together with that of a compliant CTorrent client in 15 public BitTorrent swarms. Both clients join the same torrent simultaneously to download the file and they are not rate-limited. We randomly select torrents from `www.torrentportal.com` with file size approximately between 500MB and 2GB and swarm size roughly between 50 and 650 peers. For each torrent we run the experiment only once.

Figure 2 shows that the modified free-riding client is able to download faster than the compliant client in 12 out of 15 torrents. Table I shows the downloaded file size as well as the average number of seeders and the average number of leechers in each experiment over the duration of the file download. In torrents 2, 3 and 13, the swarm size and the number of seeders are too small for the modified client to benefit. Given that the typical

Torrent#	File Size (MB)	Avg # seeders	Avg # leechers
1	538	106.20	323.82
2	738	9.88	41.21
3	683	14.19	59.14
4	955	157.42	438.16
5	782	43.41	139.61
6	889	42.29	481.18
7	697	30.83	51.89
8	873	23.12	171.43
9	1284	59.52	199.20
10	1409	242.18	341.92
11	1016	63.28	492.01
12	1146	51.35	337.21
13	1401	8.48	63.52
14	1870	91.49	392.15
15	1788	196.37	455.60

TABLE I. **The file size, and the average number of seeders and leechers over the duration of the entire file download in each public torrent experiment.**

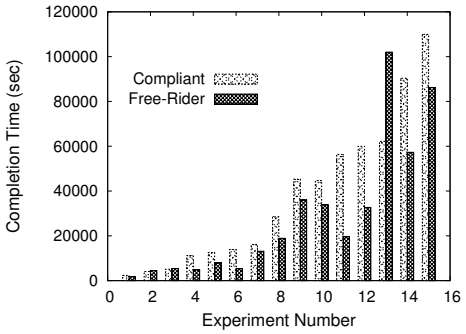


Fig. 2. **The large view exploit in public torrents.**

peer list size sent by a tracker is 50, the clients that use the exploit do not acquire a substantially larger view than compliant clients. This explains why the free-riding client does not perform better than the compliant one.

Although the swarm size of torrent 7 is small, the free-rider performs better than the compliant client because of the existence of relatively many seeders. In this case, the larger view allows free-riders to discover all the seeders in the swarm ($\sim$31). We note that it is more likely for a free-rider to download content from a seeder and for longer periods of time than it is for a free-rider to be unchoked by a leecher that possesses missing chunks. Therefore, even a small increase of the number of seeders in the client's view can yield substantial gains.

We did not observe correlation between the file size and the effectiveness of the exploit.

C. Impact of Whitewashing

With this experiment we aim at determining the impact of whitewashing with leechers (step 4 in Section III-C). As can be seen in Figure 3, free-riders that combine large view with whitewashing still outperform compliant clients in the case of 10% free-riders.

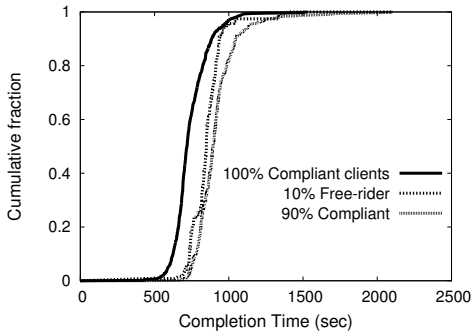


Fig. 3. **The BitTorrent large view exploit in PlanetLab-residing torrents for a 12MB file. Free-riders acquire large view *and perform* whitewashing with peers. There is one initial seeder and $\sim$300 leechers. All leechers' are rate-limited to download and upload at 30KB/sec: We depict the cases for 10% free-riders. Free-riders and compliant clients have mean download completion times equal to 852 sec and 903 sec, respectively.**

However, by comparing mean download completion times, we observe that both the compliant and free-riding clients when whitewashing *is not* employed (Figure 1(a)), outperform their counterparts when whitewashing *is* employed (Figure 3). We believe that this is because whitewashing interferes with the optimistic unchoking mechanism, thereby it prevents compliant clients from discovering cooperative clients and efficiently performing tit-for-tat data exchanges. Consequently, the rate with which content is disseminated in the swarm decreases. On the other hand, when free-riders do not employ whitewashing, cooperative clients have more opportunities to discover each other. As a result, all peers download at higher rates.

When the swarm has 1% free-riders, the free-riders that employ only large view attain better mean download completion time (644 sec, not depicted) than their counterparts that combine large view with whitewashing (680 sec, not depicted). Since the performance of both free-riding and compliant clients is adversely affected when free-riders employ whitewashing, a free-rider *is not* motivated to whitewash its leechers.

V. Addressing the Exploit

In this section, we take a step further to speculate on how to prevent the large view exploit. We propose a variation of the *verifiable pseudorandom peer selection* technique presented in [12]. Their technique assumes that each client has a certifiable public/private key pair and that clients only leave the network and never join it. In contrast, our variation takes into account the high join/leave churn of BitTorrent swarms and the impracticality of assigning certifiable public/private key pairs to BitTorrent clients.

We propose to modify the tracker and the BitTorrent

clients so that all clients have a consistent and complete view of the swarm. Each client is identified by its IP address. The tracker is synchronized with its clients and time proceeds in intervals of duration T. When a client first joins the network during interval t, it obtains from the tracker the complete view of the swarm at the end of interval $t-1$. A newly joined client does not attempt to connect to peers until interval t elapses. Clients strive to stay connected to the tracker for the duration of their download. At the end of each interval and if the client set has changed, the tracker sends to every client an update with the clients that have joined and the clients that have left the swarm during this interval. Clients do not attempt to connect to new clients right after an interval elapses. Instead, they wait for time $a << T$ until the tracker has send the updates to all clients.

Each client deterministically selects its peers using a pseudo-random number generator (common to all clients) seeded with its IP and the current time interval t. When a client A accepts a connection from a client B, it uses the same pseudorandom generator function to generate N values. Subsequently, client A maps these values to N IPs in the complete swarm view that both A and B possess. A accepts the connection if A's IP is among the resulting N IPs. Otherwise, A infers that B is a free-rider that attempts to connect to more peers than the protocol allows. In this case, the rational client or seeder A, disconnects from B.

However, it is problematic for clients to identify their peers based on the IP address. The reason is that clients that reside behind the same Network Address Translator (NAT) are allowed to connect only to the same set of peers. At the same time, most BitTorrent clients by default disallow more than one concurrent connections from the same IP, as a measure against free-riders that attempt to increase their chances of becoming unchoked. To address this issue, we do not restrict each IP to the small 50-peer swarm view with which trackers typically respond to view requests. Instead, we set $N > 50$, while compliant clients attempt to connect only to 50 peers. In this way, compliant clients behind the same NAT are able to connect to disjoint set of peers.

The proposed technique imposes increased load on the tracker and it is our future work to assess its scalability, as well as to determine good values for parameters T and a. It is also our future work to empirically determine the tunable parameter N, so that typical BitTorrent swarms maintain connectivity, while they are not susceptible to the large view exploit.

VI. Conclusion

We experimentally demonstrate a new aspect of the free-riding problem in BitTorrent. When a client obtains a larger than normal view of the BitTorrent swarm, it increases its chances to become unchoked by leechers and to discover seeders. Consequently, it is able to attain good download rates without uploading. We show that in public torrents, a free-rider can perform better than a compliant peer. We also show that in PlanetLab-residing torrents, free-riders on average outperform compliant clients, except when free-riders dominate the swarm, in which case the performance of both compliant and free-riding clients is substantially degraded.

These results suggest that selfish (rational) users may have incentive to adopt the exploit. To address this problem, we suggest a technique that enables BitTorrent clients to determine whether their newly connected peers are attempting to free-ride.

VII. Acknowledgements

We thank Nikitas Liogkas for the numerous fruitful discussions on this work. This work was supported in part by NSF award CNS-0627166.

References

[1] Bnbt easy tracker. bnbteasytracker.sourceforge.net.
[2] Enhanced ctorrent. www.rahul.net/dholmes/ctorrent/.
[3] Peer exchange. www.azureuswiki.com/index.php/Peer_Exchange.
[4] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on cooperation in bittorrent communities. In *P2PEcon*, June 2005.
[5] R. Axelrod. The evolution of cooperation. Basic Books, 1984.
[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. In *SIGCOMM CCR*, July 2003.
[7] B. Cohen. Incentives build robustness in bittorrent. In *P2PEcon*, June 2003.
[8] J. R. Douceur. The sybil attack. In *IPTPS*, March 2002.
[9] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *IEEE JSAC*, May 2006.
[10] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: The bell tolls? In *IEEE Distributed Systems Online*, June 2005.
[11] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *P2PEcon*, August 2005.
[12] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *OSDI*, November 2006.
[13] N. Liogkas, N. R., E. Kohler, and L. Zhang. Exploiting bittorrent for fun (but not profit). In *IPTPS*, February 2006.
[14] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free Riding in BitTorrent is Cheap. In *HotNets*, November 2006.
[15] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent? In *NSDI*, December 2006.
[16] J. Shneidman, D. Parkes, and L. Massoulie. Faithfulness in internet algorithms. In *PINS*, September 2004.
[17] B. P. Specification. www.bittorrent.org/protocol.html.