

MegaTorrent: An Incentive-Based Solution to Freeriding in P2P File-Sharing Networks

Peng Shi

Kshipra Bhawalkar

Abstract—We present MegaTorrent, a modification of BitTorrent that eliminates freeriding without artificially restricting system efficiency. It relies on a notion of “reputation” similar to that in the “EigenTrust” Algorithm or in Google’s PageRank: users improve their reputation by uploading to peers with high reputations, and as a result, having a high reputation becomes desirable. We prove that given rational individual behaviors, the mechanism is naturally fair—users get according to how much they give. Furthermore, the mechanism does not introduce artificial constraints such as requiring strict exchange. Megatorrent is also robust against peer collusion and multiple identity attacks. We reaffirm our theoretical findings through simulation, and demonstrate that unlike in BitTorrent, freeriders cannot hope to get by in MegaTorrent.

I. INTRODUCTION

Over the past few years, peer-to-peer (P2P) file-sharing has exploded in popularity. According to a study by CacheLogic Research, as of June 2006, over 60% of all consumer internet traffic is P2P [21], and most of this is file-sharing through systems such as BitTorrent, Kazaa, or EDonkey2000.¹ These systems provide an efficient way for users to share files with peers, and users choose these systems both because of the breadth of files available and because of the fast download speed. According to a recent study of BitTorrent, the average download speed is about 30 KB/s, which is large enough to allow users to fetch even large files in a day [13].

The Achilles heel of P2P file-sharing is Freeriding. This occurs when users use the resources of the P2P network without contributing back. Many users of P2P file sharing systems not only do not upload new files to the system, but during their downloads, they cap their upload rates at the minimum, and move the file immediately after download so they do not upload to others. As the P2P network expands, the freeriders deplete the network’s resources and drag down the download rates, making the system worse for all. Economists call this “the Tragedy of Commons” [12].

The evolution of P2P content sharing technology can be seen as sequentially better attempts to tackle the freerider problem. In the first P2P networks, freeriding was rampant. One of the first studies on freeriding considered the Gnutella system, and found that almost 70% of Gnutella users never contributed to the system, and nearly 50% of queries were served by the most magnanimous 1% of the users [1]. Today, relatively few people use Gnutella due to the slow download rates. The next generation of P2P systems, such as *Kazaa*, tried to limit freeriding by computing some type of “reputation

score” [24] through measuring the ratio of uploads versus downloads as well as the quality of files uploaded. The system then prioritizes users with high reputations. However, users hacked the clients and artificially boosted their reputations, and freeriding continued as before. A big breakthrough came with the advent of BitTorrent, which breaks files into many “chunks” and have users obtain files by trading with peers. This system now dominates P2P file-sharing, claiming about 33% of all internet bandwidth in the US [9].² In BitTorrent, each user maintains connections with 4 peers who are uploading to him/her the most, and the other peers are “choked”—temporarily dropped as trading partners. The creators of BitTorrent argue that this choking algorithm follows the ancient notion of “tit-for-tat” [6]—an eye for an eye, a tooth for a tooth—which is one of the best known strategies to the repeated Prisoner’s Dilemma. As a corollary, they claim that BitTorrent provides robust incentives against freeriding.

However, later studies showed that BitTorrent does not in fact possess sufficient incentives against Freeriding [14]. According to Jun and Ahamad [14], the problem is that choking in BitTorrent is not strictly “tit-for-tat”: users perform so-called “optimistic unchoking”—uploading chunks to a random peer for up to 30 seconds [14] to lure the peers with higher upload bandwidths to trade chunks. Jun and Ahamad showed empirically that in the current BitTorrent system, a user’s average completion time has little correlation with that user’s upload rate [14]. Qiu and Srikant studied BitTorrent’s performance both analytically and experimentally, and they found that optimistic unchoking provides opportunities to freeride [18]. These studies demonstrate that BitTorrent does not possess a robust incentive mechanism to eliminate freeriding.

Given that users act in their own self interests, there have been many attempts to construct an incentive-compatible mechanism that enforces fairness [15], [14], [6], [23], [20], [18]. Proposed schemes include peer auditing [25], upload credits and download debits [16], utility-based allocation [19], chunk-for-chunk exchange [2], [17], and strict tit-for-tat choking [14]. However, none of these solutions are satisfactory: auditing and credit systems are vulnerable to hacking or peer collusion; utility-based allocation requires central allocation, which is not scalable and is vulnerable to peer deception; exchange-based or strict tic-for-tat mechanisms give sufficient incentives against freeriding, but they hurt the average down-

¹For a survey of state of art peer-to-peer content distribution technologies, see [3].

²In China and Europe, EDonkey2000 is more popular. EDonkey also breaks files into chunks, but it does not have BitTorrent’s elaborate choking algorithm. Its big advantages over BitTorrent is better searching capabilities and availability of files.

load rate more than freeriding.

A related problem of BitTorrent is that it does not give incentives for users to become seeders—users who magnanimously upload to peers. Pouwelse et al. conducted a measurement study of BitTorrent [13] and they found that a major bottleneck is that seeds with high uptime are rare: only 17% of peers have an uptime longer than one hour after downloading; for 10 hours this number decreases to 3.1%. But seeds are very important to the system: their study finds that having a single peer with high uptime makes the difference between a content lifetime of 10 days or 100 days [13]. While discourage freeriding, we also want to encourage seeding.

A. Our Contributions

We propose MegaTorrent—a robust incentive-based approach to eliminate freeriding and encourage seeding. Instead of having users control their download rates through peer-selection as in BitTorrent, we only give users direct control over their *upload rates* to peers. Each user is assigned a score for “reputation” or “importance,” which we term the user’s “Eigenscore,” and users can increase their Eigenscore only by uploading to peers with high Eigenscores. This makes having a high Eigenscore desirable, and motivates users to upload to the network and reward the peers with high Eigenscores. In our mechanism, selfish individual actions naturally enforces fairness—it is in the best interest of each user to make sure that peers get according to how much they give.

Moreover, our way of computing Eigenscores makes MegaTorrent robust against peer collusion and multiple identity attacks. We compute them as follows: consider the peer network as a Markov chain in which each user is a node. Each user draws an out edge to each peer, and the edge weight is proportional to the discounted volume of transfers from that peer to the user, normalized so that the sum of weights of out-edges from each node is 1. The Eigenscore for each user is then the steady-state probabilities of being in the corresponding node in this Markov chain. This is analogous to Google’s PageRank [5]. Kamvar, Schlosser and Garcia-Molina studied a similar system called “Eigentrust” [22], which they used to control the spread of inauthentic files in P2P networks³. As far as we know, we are the first to use such a scheme to encourage uploads.

The organization of the paper is as follows. In Section II, we analyze currently proposed mechanisms to curb freeriding and show why none of them are satisfactory. In Section III, we list the design goals for our mechanism. In Section IV, we describe the MegaTorrent mechanism. In Section V, we analytically prove that MegaTorrent is incentive compatible and fair. In Section VI, we show that our mechanism is robust. In Section VII, we analyze our mechanism through simulation and reaffirm the properties we prove. We also compare it with BitTorrent and show that MegaTorrent provides clear incentives to upload. In Appendix A, we show how to implement MegaTorrent in a distributed and secure way, based on schemes proposed in [22].

³Other reputation systems are studied in [7], [26].

II. CURRENTLY PROPOSED MECHANISMS

A. Peer Auditing

Ngan, Wallach and Druschel considered having users audit each other’s contributions to the network through physical devices or software quota managers [25]. Although their mechanism is designed for P2P storage rather than content distribution, it is conceivable that some similar framework can work in a file sharing setting, in which users force one another to upload by controlling one another’s’ download rates. The problem is that users can bypass the auditing through hacking or collusion. Eger and Killat also study the proposal that peers should control the service rates to their neighbors [8]. These schemes use the intuition from psychology that users are willing to expend resources to punish peers. For example, in a study conducted by Fehr and Gächter, human subjects were willing to spend their own money to punish selfish peers [11]. Our Eigenscore system in fact borrows the same intuition.

B. Upload Credits and Download Debits

Gupta, Judge and Ammar considered mechanisms in which users are credited for uploading to the system and/or debited when downloading [16]. They considered 2 approaches, which they called Debit and Credit Reputation Score (DCRC) and Credit Only Reputation Score (CORC). Both schemes are fatally flawed. The DCRC mechanism is vulnerable to multiple identity attacks, in which users who have low reputation can simply create a new identity. Although it is theoretically possible to prevent multiple identities through some type of ID-ing, a system that uses virtual identification (*i.e.* IP-based) we be hacked sooner or later, and a system that uses a real identification (*i.e.* Driver’s License or SSN) will compromise the anonymity appeal of P2P file-sharing (most of which is for illegal content). The CORC mechanism is vulnerable to peer collusion, in which two peers add to each other’s credit without paying any cost.

C. Utility-Function based allocation

Ramaswamy and Liu examine utility function based schemes for allocating P2P resources [19]. They consider a variety of utility functions and give users incentives to improve their *utility score* to the system. However, they only considered a theoretical framework without a specific scheme. They do not provide an incentive scheme that lure users to play along with their system.

D. Strict Exchange Based Mechanisms

Anagnostakis and Greenwald propose a strict exchange-based mechanism in which users trade chunks of files by forming n -cycles [2]. Gauthier, Bershad and Gribble propose a similar mechanism [17]. Although these exchange-based mechanisms solves freeriding because one *has to* upload a chunk in order to get a chunk, they unnecessarily restrict the total throughput of the system. Having to find such exchange-cycles is difficult and causes a significant bottleneck. Moreover, sometimes such exchange-cycles cannot be found so network resources is under-utilized. Recall that the reason

freeriding is undesirable is that it slows down the system, so overly-restrictive mechanisms only replace freeriding with a greater evil. Gauthier et al. simulated their exchanged-based mechanism and compared with the current mechanism. For small files, their average completion time is 5 days and the median is 18 hours, compared with an average of 30.13 hours and median of 19.6 minutes in bittorrent. For larger files, their average completion time is 20.5 days and median is 21 days, compared with the current average of 4.82 days and median of 24.35 hours. This extra lag of 20-60 times render such systems acceptable.

E. Strict Tit-for-Tat Choking

Jun and Ahamad propose a modification of BitTorrent in which the choking algorithm is a stricter manifestation of Tit-for-Tat [14]. They propose that each user should keep an *upload deficit* for each peer, calculated based on the difference between the number of chunks given to the peer minus the number of chunks obtained from the peer. Choking is done when and only when the upload deficit for a peer exceeds a constant number of chunks and there is no optimistic unchoking. They showed through simulation that in this system, there is a clear downward curve of completion time versus upload rate: users who upload more download more quickly. However, their system seriously compromises network efficiency, because without optimistic choking constantly exploring new connections, their system settle on sub-optimal peer networks. According to their simulation, the average completion time in their system is 1.5 times longer than the current BitTorrent system. As with exchange based mechanisms, they traded freeriding for an arguably worse handicap to the network.

III. DESIGN GOALS FOR OUR MECHANISM

We seek to eliminate freeriding while preserving properties of current P2P mechanisms that are instrumental to their success—properties such as efficiency and robustness. We list the following design goals:

- 1) *Efficiency*: We should not artificially constrict network bandwidth with rules such as strict exchange or strict tit-for-tat.
- 2) *Fairness*: Each user gets a share of the resources available in the P2P network roughly weighted by its contribution [8]. The amount of download of each user should roughly equal the amount of uploads.
- 3) *Incentives to Seed*: The system should give incentives for users to remain as seeders after their download has completed. This will vastly prolong the time files remain in the system [13].
- 4) *Robustness*: The incentive scheme should not be distorted by attacks such as peer collusion or multiple identities.
- 5) *Practicality*: The mechanism should be able to be implemented in a scalable, distributed, secure way.

We present an new mechanism that is efficient because we build it on top of BitTorrent and impose no restrictive additional condition. We prove that it is fair (Corollary V.4)

in Section IV. We argue that it is robustness in Section VI and that it is practically implementable in Appendix A. We reaffirm our arguments with empirical tests in Section VII.

IV. OUR MECHANISM

MegaTorrent relies on a type of reputation score called Eigenscore. One can only increase one’s Eigenscore by uploading files to peers with high Eigenscores. This is done through having users rewards peers for uploading to the user by giving peers recommendations, and after normalizing the sum of recommendations from each user to 1, we calculate the Eigenscore by computing the first eigenvector of the recommendations graph. We define this more precisely in Section IV-B. This system is similar to Google’s PageRank [5] and the “Eigentrust” system by Kamvar, Schlosser and Garcia-Molina [22].

As in BitTorrent, files are broken into chunks, and there are trackers and websites with .torrent files. To download a file, the user contacts a tracker as in BitTorrent and obtain a list of peers possessing the file, and sends requests to peers in a random order. In contrast to BitTorrent, MegaTorrent leaves the decision of whether or not to fulfill a request completely to the uploader; the uploader decides how much (if any) uploading bandwidth is allocated to each requester, and this decision is based on the uploader’s desire to maximize his/her Eigenscore.

Each user will want to maximize their Eigenscore because peers will reward higher Eigenscore with more transfers; in so doing each user helps to enforce the fairness of the system. We rigorize these ideas in Section V.

For simplicity, we first describe our mechanism using continuous variables and known parameters. We show how to implement such an algorithm practically in Appendix A.

A. Notation

Suppose there are n users in the P2P system and each user is referred to by an index. At time t , define the following quantities:

- 1) $T_{ij}(t)$: the instantaneous rate of transfer from user i to peer j at time t .
- 2) γ : a system-wide discount factor. This is a positive real parameter that can be tuned so that $1/\gamma$ roughly corresponds to the time horizon that we want remember historic contributions. This is needed to make Eigenscores numerically stable.
- 3) $H_{ij}(t) = \epsilon + \int_0^t T_{ij}(t-s)e^{-\gamma s} ds$. This is the discounted total volume of transfers from peer i to j , up to date at time t . The addition of constant $\epsilon > 0$ makes sure that the graph used to compute Eigenscore is strongly connected.
- 4) $H_j = \sum_{i \neq j} H_{ij}$. This is the total discounted volume of downloads of user j . This measures how much resources user j is obtaining from the network and takes into account how recent it is.

- 5) $G_j = \sum_{i \neq j} H_{ji}$. This is the total discounted volume of uploads of user i . This measures how much user i has contributed to the network.
- 6) $I_j = p_j/H_j$ is called the priority index for user j . We show that users prefer to upload to the peers with the highest I_j and in so doing enforce fairness.

User i only has control over his/her uploads H_{ij} .

B. Reputation Measure: Eigenscore

Definition 1. For any time t , define a Markov chain with n nodes, such that there is a transition edge from i to j with rate $H_{ij}(t)/H_j(t)$. This graph is strongly connected because we stipulated $H_{ij}(t) \geq \epsilon > 0$, so the corresponding Markov chain is ergodic. Define $p_i(t)$ as the steady-state probability of being in node i . We call $p_i(t)$ the Eigenscore for user i at time t .

Lemma IV.1. The Eigenscore p 's satisfy the equation

$$p_i(t) = \sum_{j \neq i} p_j \frac{H_{ij}(t)}{H_j t}$$

Proof: This follows from flow conservation in the steady state of the recommendation Markov chain. ■

The Eigenscores \vec{p} can be computed simply by taking sequentially higher powers of the recommendation matrix A with entries $\langle a_{ij} = H_{ij}/H_j \rangle$ and multiplying by an arbitrary positive unit vector \vec{c} . This converges very quickly to the final \vec{p} in practice, as shown in Figure 1, which uses an Eigenscore with 25 users taken from our simulation.

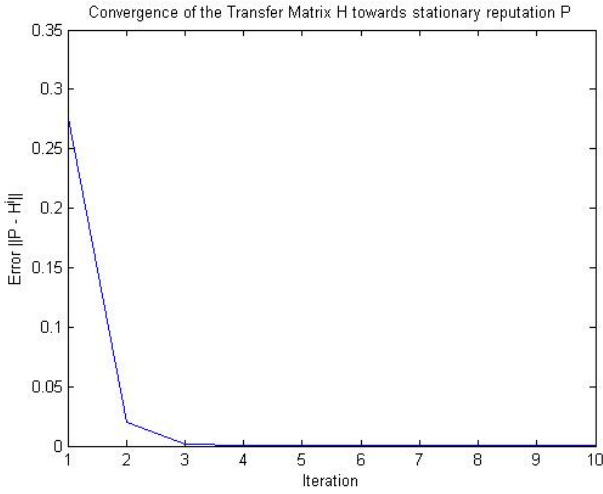


Fig. 1. Plot of $\|\vec{p} - A^i \vec{c}\|$ versus i . The computation converges very quickly.

V. INCENTIVE COMPATIBILITY

We proceed to show that our mechanism is incentive compatible: it is in each user's best interest to enforce fairness in the system. We assume here that users truthfully report their H_i 's and give proper recommendations to peers. We relax these conditions in Appendix A.

A. Individual Eigenscore Maximization

Recall the notations from Section IV-A. Since users can only increase their download rate by increasing their Eigenscore (this is reaffirmed by Corollary V.2), rational users will maximize their Eigenscore subject to a personally determined upload bandwidth cap L_i :

$$\text{Maximize } p_i(t) = \sum_{j \neq i} p_j(t) \frac{H_{ij}(t)}{H_j(t)} \quad (\text{EIGENSCORE MAX.})$$

$$\begin{aligned} H_{ij}(t) &= \int_0^\infty T_{ij}(t-s)e^{-\gamma s} ds \\ \sum_{j \neq i} T_{ij} &\leq L_i \\ T_{ij}(t) &\geq 0 \end{aligned}$$

We also introduce the following assumption.

Assumption 1. At any time instant t , user i cannot significantly alter the reputation of user j by altering i 'th own upload amounts $H_{ik}(t)$. i.e. $\frac{\partial p_i(t)}{\partial H_{ik}(t)} \approx 0$. Moreover, user i cannot significantly alter the upload decisions of user j . i.e. $\frac{\partial H_{jl}(t)}{\partial H_{ik}(t)} \approx 0 \forall k, l$.

Assumption 1 is justified because p_j is mostly controlled by user j 's choice of upload amounts H_{jk} 's, which user i cannot directly control. Although user i can theoretically change his/her upload amounts T_{ik} 's to alter the system wide p_k 's to induce a change in behavior in user j , in a real network, no user has the upload bandwidth to game the whole system in such a way.

Theorem V.1. To maximize his/her own eigenscore, user i will upload to the peers with the highest $I'_j = p_j(H_j - H_{ij})/H_j^2$.

Proof: Consider the individual optimization problem for user i at time instant t . Because the user can only control H_{ij} . We view the objective p_i as a function of H_{ij} (at time t) where $1 \leq j \leq n$, $j \neq i$. We wish to maximize

$$p_i = \sum_{j \neq i} p_j(t) \frac{H_{ij}(t)}{H_j(t)}$$

Subject to $\sum_j T_{ij} \leq L_i$. By assumption 1, $\frac{\partial p_j}{\partial H_{ij}} = 0$, and $\frac{\partial H_{kj}}{\partial H_{ij}} = 0$ for all k . This implies that $\frac{\partial H_j}{\partial H_{ij}} = \frac{\partial H_{ij}}{\partial T_{ij}} = 1$. Using these expressions, we take the partial derivative of p_i with respect to H_{ij} ,

$$\frac{\partial p_i}{\partial H_{ij}} = p_j \frac{H_j - H_{ij}}{H_j^2}$$

Given the constraint $\sum_j T_{ij} \leq L_i$, and since $\frac{\partial H_{ij}}{\partial t} = T_{ij}$, the optimal allocation is to choose the peer j that requires a file with the highest $I'_j = p_j \frac{H_j - H_{ij}}{H_j^2}$ and upload as quickly as possible to that user, then use any remaining upload bandwidth to upload as quickly as possible to the user with the second highest I'_j and so on. ■

Corollary V.2. All else being equal, each user will prefer to upload to peers j with high p_j .

In order to gain further intuition, we simplify the expression in Theorem V.1 by adding the assumption that users tend to download from multiple peers. *i.e.* $H_{ij}(t) \ll H_j(t)$. This is reasonable because users come online at different times and they are interested in different files, so it makes sense that out of all past downloads of user j , only a relatively small portion come from user i .

Assumption 2. For any i, j , $H_{ij}(t)/H_j(t) \ll 1$.

Theorem V.3. *The rational action for user i is to upload as quickly as possible to the users with the highest $I_j = p_j/H_j$, prioritizing peers with higher I_j 's but staying within a total upload bandwidth cap of L_i . Hence, the system naturally makes the I_j 's approximately equal to some global I .*

Proof: When $H_{ij}(t)/H_j(t) \ll 1$, the expression in Theorem V.1 $I_j' = p_j \frac{H_j - H_{ij}}{H_j^2}$ converges to $I_j = p_j/H_j$. So the first part of the corollary follows from Theorem V.1.

For the second part, note that when each user i upload as quickly as possible to peer j with a high $I_j = p_j/H_j$, so H_j increases, and I_j decreases. (This is not cancelled out by possible increases in the numerator because $\frac{\partial p_j}{\partial H_{ik}} \approx 0$ by Assumption 1.) Since all rational users will follow this strategy, the higher I_j 's will decrease, and the system will tend toward an equilibrium in which the I_j 's are equal. Define this equilibrium value as I . (Because of the dynamism of the system, the random times of requests, and the various limits to bandwidth, one should not expect a precise equilibrium to occur in practice. Rather, in the long run, the I_j 's will be approximately equal.) ■

Corollary V.4. *The system tries to make $G_i(t) \approx H_i(t)$ for each i .*

Proof: Using Theorem V.3 and Lemma IV.1,

$$\begin{aligned} H_i(t) &= p_i(t)/I_i(t) \\ &\approx \frac{p_i(t)}{I(t)} \\ &= \sum_{j \neq i} \frac{p_j(t)}{I(t)} \frac{H_{ij}(t)}{H_j(t)} \\ &\approx \sum_{j \neq i} H_j(t) \frac{H_{ij}(t)}{H_j(t)} \\ &= \sum_{j \neq i} H_{ij}(t) = G_i(t) \end{aligned}$$

This makes the system naturally fair: it is in each user's own interest to enforce that the amount of downloads of each user is roughly equal to the amount of uploads by that peer. This enforcement does not introduce artificial network restrictions and it always encourages users to upload as much as possible. Thus, our mechanism results in a better social outcome than current and currently proposed mechanisms.

VI. ROBUSTNESS OF MEGATORRENT

Of course, the robustness of MegaTorrent depends on the truthful computations of Eigenscores and upload statistics. The problem of computing such values in a distributed, secure way has been extensively studied [22]. For details of how we apply known techniques to our mechanism, see Appendix A. For now, we assume that we can accurately calculate the Eigenscores p_i 's and download volume H_i 's. We show that MegaTorrent is stable when users join or leave, and safe against peer collusion and multiple identity attacks.

A. Users Joining

When a new user i joins the system, the user's Eigenscore p_i is essentially 0. (It is not exactly 0 because we initialize $T_{ij} = \epsilon > 0$.) Such a user cannot download anything. In fact, new users in our mechanism have to first contribute and "prove their worthiness". After registering an account, the user will receive some random files from a central server, and must become a seeder for a while to build up a positive Eigenscore p_i . But our mechanism is fast to respond: as soon as the user builds up some reputation, he/she will be able to download because he/she will have a large index $I_i = p_i/H_i$ (since the denominator $H_i \approx 0$), and peers will honor the user's requests. Having new users to contribute first is reasonable because normal users should be willing to put in a one-time contribution in order to reap the immense reward of being part of a P2P network.

B. Graceful Exits

When a user i stops using the system for a long time, his/her Eigenscore p_i will decay exponentially with time, because $H_{ij}(t)$ is discounted exponentially. A user who stops using the system will not disrupt the system. Rather, his/her presence in the network will be reduced until no trace remains.

C. Peer Collusion

Unlike in a naive credit-based system, freeriders cannot boost their reputations by falsely recommending one another. This is handled implicitly by our PageRank-type algorithm for calculating Eigenscores: unless users increase their Eigenscores by legitimately uploading to peers, their recommendations do not have any weight.

D. Multiple Identity Attacks

Users cannot gain by making new identities, because new users have reputation values of essentially 0. Since Eigenscores are always positive, new users have the worst possible status, and it does not help a freerider to start a new account. This simply annuls any contribution the user had done in the past and forces the user to rebuild reputation from scratch.

VII. SIMULATION

A. Our Simulator

Although there are many available simulators for P2P systems, (a partial survey and comparison has been conducted by Naicken et al [10]), it was very difficult finding one

that suits our needs. Therefore, we decided to code our own from scratch. For simplicity sake, we ignored the low-level protocols, but simulated a clean, round-based model that captures the key elements of our mechanism.

The central unit in this simulation is a clearing house that conducts multiple clearing rounds. In each round the clearing house fetches demands from all nodes. These demands specify the file that the node wishes to obtain. The clearing house then compiles demands from all nodes and uses some protocol dependent algorithm to select a subset of demands to fill in the given round. Of course, no physical transfer takes place. Nodes are only marked to have sent or received a file. To allow fairness in the system nodes go in a random order while selecting demands to fill from the global set of demands.

There are some constraints imposed on the system. Each node has a upload capacity and a download capacity which specifies how many files a node can upload or download in a given round. Moreover each link from a node to another (we assume completely connected bidirectional graph) has a bound associated with it. This bound specifies how many files can be transferred from one end to another in a round. Some additional protocol specific constrains may apply.

For MegaTorrent the demand selection protocol selects nodes based on the value $p_j(H_j - H_{ij})/H_j^2$ where j is the node making the demand. The Demand Selection scheme for BitTorrent is more complicated due to the presence of optimistic unchoking: In each step two peers that are demanding chunks are chosen, based on the number of chunks that were provided by them in the last round. Additionally one more peer is selected randomly to upload to. This random selection mechanism is called optimistic unchoking. In each case, upload capacity permitting all demands from nodes are met.

Our simulator is simple and simulates only a small part of the general skeleton of a peer-to-peer network. We believe that despite its simplicity the simulator provides a fair representation of activities in a general peer-to-peer network. Currently in the simulator all the decisions are made using complete information. A side-effect of this is that the simulator is a bit slow and our experiments have been limited to a small networks. We plan to make the simulator more distributed in the future.

In the next two subsections we describe some of the results obtained using the simulator.

B. Verifying Properties of MegaTorrent

To verify our theoretical results, we examined how the download rate and reputation (Eigenscore) of a user changes as his/her upload capacity changes. The first plot in figure 2 shows the relation between reputation and upload capacity. The reputation (Eigenscore) increases as upload capacity increases thus a user who wants a higher reputation will need to increase his upload capacity. In the second plot in figure 2 we plot download rate in relation to upload capacity. Download rates are computed by counting the average number of files downloaded in each round. Based on the graph MegaTorrent

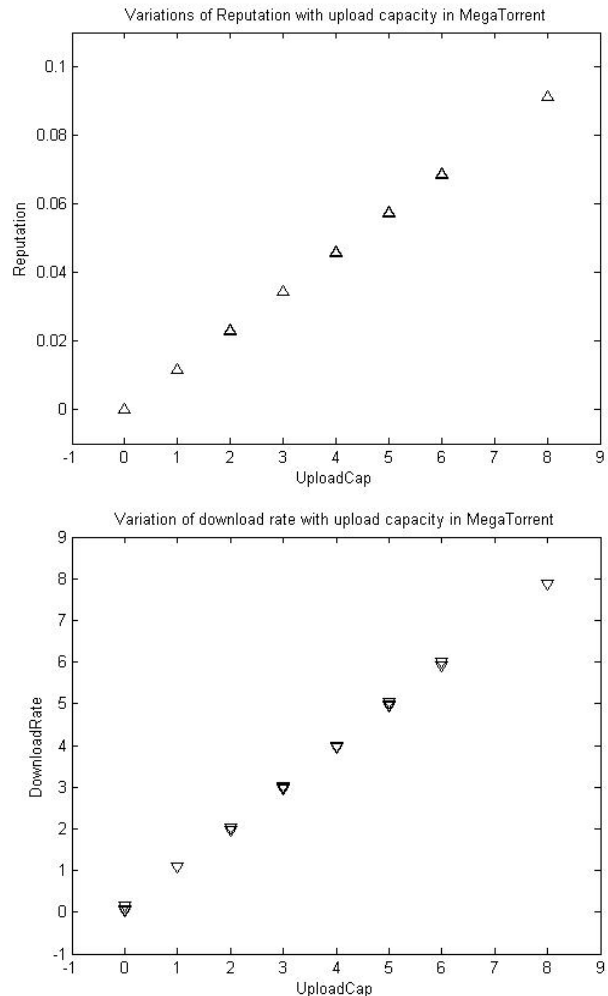


Fig. 2. Nodes with high upload cap enjoy higher download rate and reputation while nodes with low upload cap suffer from low download rate and reputation. Note that the relations in both graphs are linear as expected from Corollary V.4.

users with high upload capacity also enjoy high download rate. The perfectly linear plots reaffirm Corollary V.4.

C. Comparison with BitTorrent

Next we compare how MegaTorrent compares to BitTorrent. We tracked how the download rate varies as a single user changes his/her upload capacity, fixing everyone else. We observed that in MegaTorrent the download rate enjoyed by the user increases with upload cap while in BitTorrent it does not. This indeed shows that unlike in BitTorrent, users in our system are given clear incentives to set a high upload cap. Moreover, the graph shows for those who provide much uploads to the system, MegaTorrent will provide download rates much higher than what can be obtained in BitTorrent.

VIII. FUTURE WORK

We indicate the following directions for future research.

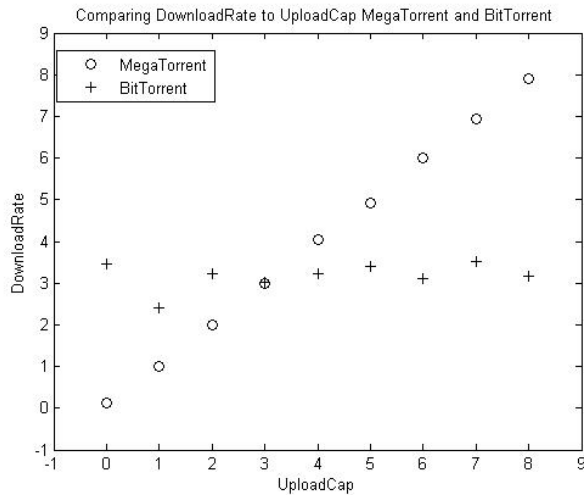


Fig. 3. Comparison of download rates in MegaTorrent and BitTorrent of a single user, under identical parameters and peer configuration. The circles correspond to MegaTorrent protocol and the +’s correspond to BitTorrent. Note that in MegaTorrent the average download rate increases with upload capacity while in BitTorrent it does not.

- 1) Measure the computational overhead of implementing MegaTorrent in the distributed, secure fashion as outlined in Appendix A.
- 2) Examine the assumptions in Section V more closely or prove analogous results with relaxed assumptions.
- 3) Study MegaTorrent using a more realistic simulator.
- 4) Implement MegaTorrent in full functionality and distribute it online. Conduct measurement studies of how the system functions compared to BitTorrent or EDonkey.

IX. CONCLUSION

We examined past attempts to tackle the freerider problem in P2P file-sharing networks, and found that all currently proposed mechanisms that ensure fairness are either not robust or overly-restrictive; they are either vulnerable to simple attacks or unacceptably detract from the system’s overall performance. But the original problem with freeriding is that it restricts the total throughput of the system, so such solutions would only replace freeriding with a greater evil. We presented MegaTorrent—an incentive compatible, robust mechanism that ensures fairness without introducing barriers to efficiency. We analytically proved that individual selfish behavior in this mechanism induces an outcome that is fair and efficient: the mechanism naturally enforces the constraint that users can only get from the network roughly how much they gave; users are also given incentives to seed as much as they can. We reaffirmed these results through simulation, and showed that while freeriders can still download at a reasonable rate in BitTorrent, they cannot get by in MegaTorrent. The next step would be to implement MegaTorrent in full functionality and to study its performance in practice.

REFERENCES

- [1] E. Adar and B.A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.
- [2] Kostas G. Anagnostakis and Michael B. Greewald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems*, 2004.
- [3] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.
- [4] S.D. Kamvar B. Yang and H. Garcia-Molina. Secure score management for p2p systems. Technical report, Stanford University, 2003.
- [5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hyper-textual web search engine. *Computer Networks and ISDN Systems*, 30, 1999.
- [6] Bram Cohen. Incentives build robustness in bittorrent. www2.sims.berkeley.edu/research/conferences/p2pcon/papers/, 2003.
- [7] S. C. Vimercati E. Damiani and S. Paraboschi. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–212, 2002.
- [8] Kolja Eger and Ulrich Killat. Fair resource allocation in peer-to-peer networks. *Computer Communications*, 30, November 2007.
- [9] Ernesto. Bittorrent: The “one third of all internet traffic” myth. <http://torrentfreak.com/bittorrent-the-one-third-of-all-internet-traffic-myth/>, 2006.
- [10] S. Naicken et al. The state of peer-to-peer simulators and simulations. *ACM SIGCOMM Computer Communication Review*, 37(2):95–98, 2007.
- [11] E. Fehr and S. Gächter. Altruistic punishment in humans. *Nature*, 415:137–140, January 2002.
- [12] N.S. Glance and B.A. Huberman. Dynamics of social dilemmas. *Scientific American*, March 1994.
- [13] D.H.J. Epema J.A. Pouwelse, P. Garbacki and H.J. Sips. A measurement study of the bittorrent peer-to-peer file-sharing system. Technical report, Delft University of Technology, 2004.
- [14] Seung Jun and Moustaque Ahamad. Incentives in bittorrent induce free riding. In *SIGCOMM '05: Proceedings of the 2005 Conference of the Special Interest Group on Data Communication*, pages 116–121, 2005.
- [15] Ion Stoica Michal Feldman, Kevin Lai and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: The fifth ACM Conference on Electronic Commerce*, pages 102–111, 2004.
- [16] Paul Judge Minaxi Gupta and Mostafa Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 144–152, 2003.
- [17] Brian Bershad Paul Gauthier and Steven D. Gribble. Dealing with cheaters in anonymous peer-to-peer networks. Technical report, University of Washington, January 2004. Available at www.cs.washington.edu/homes/gribble/papers.
- [18] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 Conference of the Special Interest Group on Data Communication*, pages 367–377, 2004.
- [19] Lakshmesh Ramaswamy and Ling Liu. Free riding: A new challenge to peer-to-peer file sharing systems. In *HICSS '03: Proceedings of the 36th Hawaii International Conference on System Sciences*, 2002.
- [20] Rahul Telang Ramayya Krishnan, Michael D. Smith. The economics of peer-to-peer networks. *JITTA: Journal of Information Technology Theory and Application*, 5(3), September 2003.
- [21] CacheLogic Research. P2p file sharing—the evolution distribution chain. <http://www.dcia.info/activities/p2pmswdc2006/ferguson.pdf>, 2006.
- [22] Mario T. Schlosser Sepandar D. Kamvar and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW2003: Proceedings of the Twelfth International World Wide Web Conference*, pages 640–651, 2003.
- [23] Jeffery Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In *IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-peer Systems*, pages 139–148, 2003.
- [24] Kazaa Team. Kazaa participation level. <http://www.kazaa.com/>.
- [25] Dan S. Wallach Tsuen-Wan Ngan and Peter Druschel. Enforcing fair sharing of peer-to-peer resources. In *IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-peer Systems*, 2003.

- [26] Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *P2P '03: Proceedings of the Third International Conference on Peer-to-Peer Computing*, 2003.

APPENDIX

We resolve various issues with implementing MegaTorrent in practice.

A. Discretizing Parameters

To actually implement MegaTorrent, we do not want to deal with the integrals in Section IV-A, but we want discrete quantities. To this end, we pick a time interval Δ and redefine $T_{ij}(t)$ as the volume of transfers from i to j in within time interval $(t, t + \Delta)$. We then let $\alpha = e^{\gamma\Delta}$ and redefine $H_{ij}(t) = \epsilon + \sum_{i=0}^{\infty} T_{ij}(t - i\Delta)\alpha^i$. The actual algorithm is as before, and the same theorems can be proven completely analogously.

B. Distributed, Secure Eigenscore

The task of computing PageRank-like reputations in a distributed, secure fashion is a solved problem. Kamvar, Schlosser and Garcia-Molina studied a reputation system completely analogous to our Eigenscore in [22], except they constructed the recommendation graph differently. They described a distributed way of computing Eigenscore. Moreover, by having multiple peers compute the Eigenscore for each users, they designed a secure way of computing the Eigenscores in which no user can tamper with his/her own score. The actual algorithm uses a Distributed Hash Table and is described in detail in [22]. Yang, Kamvar and Garcia-Molina develop on their algorithm in a more general context in [4].

C. Distributed, Secure Download Statistics

MegaTorrent also requires accurate knowledge of user i 's historical download volume H_i . To do this, we can use the general machinery for managing some kind of score presented in [4], which uses a Distributed Hash Table for distributed, secure score computation.

A simpler approach is to trust users to compute their H_i themselves but audit random users, To audit, we can ask each peer j to submit H_{ji} and sum to check if this equals H_i . We can then ban users who distort their H_i .

Moreover, based on the observation that users will willingly punish selfish peers (see Section II-A), users will be motivated to give correct recommendations. We can also audit this randomly and punish malicious users.