CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
*Concurrency Computat.: Pract. Exper.* 2013; **25**:899–917
Published online 3 May 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2856

# The problem of upload competition in peer-to-peer systems with incentive mechanisms

M. Meulpolder [1,*,†], L.E. Meester [2] and D.H.J. Epema [1]

[1] *Department of Software Technology, Delft University of Technology, Mekelweg 4, 2628 CD Delft, Netherlands*
[2] *Department of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, Netherlands*

SUMMARY

As peer-to-peer (P2P) file-sharing systems revolve around cooperation, the design of upload incentives has been one of the most important topics in P2P research for more than a decade. Several deployed systems, such as private BitTorrent communities, successfully manage to foster cooperation by banning peers when their *sharing ratio* becomes too low. Interestingly, recent measurements have shown that such systems tend to have an *oversupply* instead of an undersupply of bandwidth designers that have been obsessed with since the dawn of P2P. In such systems, the 'selfish peer' problem is finally solved, but a new problem has arisen: because peers have to keep up their sharing ratios, they now have to compete to *upload*. In this paper, we explore this new problem and show how even highly cooperative peers might in the end not survive the upload competition. On the basis of recent measurements of over half a million peers in private P2P communities, we propose and analyze several algorithms for *uploader selection* under oversupply. Our algorithms enable sustained sharing ratio enforcement and are easy to implement in both existing and new systems. Overall, we offer an important design consideration for the new generation of P2P systems in which selfishness is no longer an issue. Copyright © 2012 John Wiley & Sons, Ltd.

Received 26 August 2011; Revised 23 December 2011; Accepted 14 April 2012

KEY WORDS: peer-to-peer; incentives

## 1. INTRODUCTION

As peer-to-peer (P2P) file-sharing systems only work when a significant part of the population contributes, it is widely recognized that peers should be motivated either technically or socially to provide content to the network. As a result, a significant fraction of the research in P2P file-sharing systems has focused on the problem of providing *sharing incentives* to prevent free riding. Over the years, many incentive mechanisms have been proposed with varying degrees of decentralization [1–3]. The currently most successful incentive mechanism is *sharing ratio enforcement* (SRE), which is widely deployed in *private BitTorrent communities* and analyzed in various studies [4–6]. With SRE, a minimum per-peer sharing ratio (the ratio between the amounts of data uploaded and downloaded) is enforced by a central tracker. With over 24 million active users, private BitTorrent communities are responsible for the most significant part of all the BitTorrent activity in the world [5].

In swarm-based P2P file-sharing systems such as BitTorrent, a swarm is a group of peers that is sharing/downloading a particular file. In a swarm, a *seeder* is a peer that is sharing a full copy of the file for free. Recent measurements [4,5] have demonstrated that swarms in private communities tend to have relatively very high numbers of seeders, which is a consequence of peers trying to keep up their sharing ratios in order not to be banned from the community. However, with the very successful

*Correspondence to: M. Meulpolder, Department of Software Technology, Delft University of Technology, Mekelweg 4, 2628 CD Delft, Netherlands.
†E-mail: michel.meulpolder@gmail.com

SRE incentive mechanism in effect, the interesting new problem of *upload competition* arises: as most files are seeded by huge numbers of seeders and the number of connections of newly arriving downloaders is limited, seeders have to compete for which of them gets to upload. Although some peers that are willing to share content may obtain unnecessarily high sharing ratios, other peers that are equally willing to share may get banned because their sharing ratios drop below the minimum value; for instance, in systems with heterogeneous upload capacities, this can easily happen to slow peers. A way to solve the upload competition problem is to have an appropriate *uploader selection* algorithm in place that is used by downloaders to select uploaders to prevent unjustified banning. In this paper, we show that the random uploader selection[‡] as applied by current P2P protocols such as BitTorrent does not solve the upload competition problem, and we present, analyze, and simulate a number of uploader selection algorithms that do.

There is an inherent trade-off in the design of SRE mechanisms. As the goal of such a mechanism is to increase the download performance in the community by motivating peers to share their content, a solution that will work in the short term is to favor the fastest peers as uploaders and seeders. However, in the long term, if peers that initially do not share (*free riders*) become seeders, the larger number of seeders in the system also increases the performance, even if these peers have low upload capacities. This trade-off between short-term and long-term alternatives should be made by community managers as different communities may have different objectives. However, we will show that current mechanisms (e.g., as employed by BitTorrent) favor fast peers as a result of the randomization employed in the algorithm involved. We argue that it should be possible to implement SRE mechanisms that target only *real* free riders, regardless of their upload capacity, and at the same time prevent (and possibly compensate) unintended effects of randomization that lead to systematic banning of certain groups of cooperative peers. We call such SRE mechanisms *sustainable*. The trade-off between short-term and long-term performance benefits is then determined explicitly by community managers instead of implicitly by randomization effects.

The contributions of this paper are as follows:

1. We identify the hitherto unexplored problem of upload competition, and we argue that in addition to a successful incentive mechanism, an appropriate *uploader selection* algorithm to be employed by downloaders needs to be in place to enable sustainable SRE.
2. We demonstrate that with a naive random uploader selection algorithm (as is currently common), in systems with heterogeneous bandwidths, it is inherently impossible for slow peers to maintain high sharing ratios.
3. We propose and analyze three sustainable uploader selection algorithms that solve the upload competition problem and that are easy to implement in practice.

In the remainder of this paper we first give an overview in Section 2 of the transition from undersupply to oversupply in P2P systems. In Section 3, we present our sharing ratio model. In Section 4, we analyze the standard random uploader selection algorithm and derive a necessary condition for sustainability. In Section 5, we propose and analyze three sustainable uploader selection algorithms that solve the problem of upload competition. In Section 6, we discuss the security aspects of our algorithms. In Section 7, we discuss related work, before presenting our conclusions in Section 8.

## 2. UNDERSUPPLY VERSUS OVERSUPPLY

In this section, we describe the transition from the classic scenario of *undersupply* of bandwidth in P2P systems to the new scenario of *oversupply* because of SRE as observed in, for example, private BitTorrent communities.

[‡]Note that the so-called *seeding strategies* are the opposite of what we discuss here: although such strategies determine how a *seeder* makes a selection of *downloaders* when there is undersupply, *uploader strategies* determine how a *downloader* makes a selection of *uploaders* when there is oversupply.



DOI: 10.1002/cpe

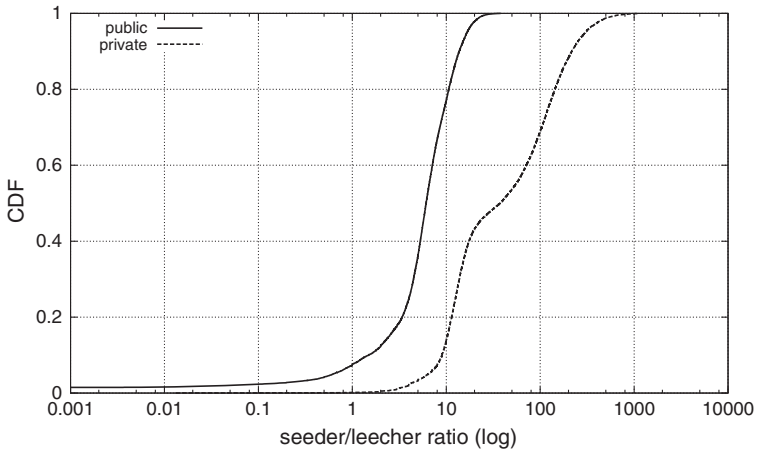


Figure 1. The CDF of the seeder/leecher ratio observed during measurements of over half a million peers in five public and private BitTorrent communities [4].

### 2.1. Sharing-ratio enforcement

Ever since the dawn of P2P, the problem of free riding has received very much attention. A classic early study on Gnutella demonstrated that nearly 70% of the Gnutella users did not share any files [7]. On the basis of these and other results, many researchers have proposed simple as well as complicated mechanisms to prevent free riding and to encourage sharing, for example, [1, 2, 8, 9]. Often, this work has been driven by the desire to solve the problem in a decentralized way—theoretically more beautiful but practically often less feasible. Although most angles of research in this area have by now been exploited, the solution that is most successful in practice is strikingly simple: centralized SRE. The *sharing ratio* of a peer is defined as the total amount of data it has uploaded divided by the total amount of data it has downloaded. To induce cooperative behavior, a minimum sharing ratio is enforced, which usually takes a value between 0.4 and 0.6 (see [4] for some examples). Usually, a peer that dips below the threshold is given a few warnings and some time to get back to an acceptable ratio before it is permanently banned. To allow new peers to bootstrap their sharing ratio, a *grace period* is common: an initial period of time during which the sharing ratio of the peer is not yet enforced.

The so-called *private BitTorrent communities* have gained prominence in the world of P2P file-sharing systems over the last few years, which may in large part be due to the fact that many of them employ an SRE mechanism [4–6]. Measurements of Zhang *et al.* [5] show that private BitTorrent communities host an estimated 4.4 million torrents and have over 24 million active peers. In such communities, each user needs a (typically hard to obtain) user account, a central tracker keeps a precise administration of the upload and download activity of each peer, and an SRE mechanism is used to ban users who are (or sometimes, as we will show, *seem*) not cooperative enough.

As expected, SRE turns out to be highly effective. In earlier work [4], we have presented detailed measurement results of over half a million peers in public and private communities, and Figure 1 shows the CDF of the seeder/leecher[§] ratio that was observed in these communities. In the private communities, only 13% of the observations indicate ratios of less than 10, whereas 30% of the observations indicate ratios exceeding 100. In contrast, as many as 80% of the observations of public communities show a ratio of less than 10. Hence, we can safely state that in private communities with SRE, the incentive problem has been sufficiently solved, and free riding is no longer an important issue.

[§] In BitTorrent language, a *leecher* is a downloader that has not yet finished downloading.



DOI: 10.1002/cpe

### 2.2. Credit mechanisms

Another way to enforce cooperation is by means of *credits* or *tokens*, such as the mechanisms proposed in [3, 10–12]. With a credit mechanism, a peer earns credit by uploading and spends credit by downloading. Commonly, a peer starts with an initial amount of credit and needs a positive balance to download. The price per unit upload is not necessarily the same as the price per unit download; by introducing asymmetry in these prices, community designers might attempt to correct certain economic effects such as credit not being used by inactive users, and users having huge amounts of credit in their accounts. In some communities in which a user can only obtain an account by invitation, his initial amount of credit is donated by the inviting user. This creates a stronger social obligation to cooperate and reduces the effects of *whitewashing*, which is the phenomenon of a user creating a new account when he has a negative balance in his old account. This paper is written from the perspective of SRE, but our analysis and conclusions naturally apply to credit mechanisms as well.

### 2.3. The new problem: upload competition

We consider a system *sustainable* if (i) peers have an incentive to be cooperative *and* (ii) cooperative peers are able to stay in the system without being unfairly banned. A system with SRE provides the incentive of (i). However, to satisfy (ii), it is required as well that a peer's sharing ratio should *only* drop below the threshold if it refuses to contribute. Both the setting of the threshold and the upload/download protocol influence the sustainability of a system.

When a strict sharing ratio minimum or credit minimum is enforced, each user has to make sure to stay above the threshold. However, the means of a user to achieve this are limited: a user merely has the choice of which files to make available to possible downloaders, but whether downloaders will actually appear *and* download from this particular user is beyond his control. Here lies the crux of a new problem: users might experience a decreasing sharing ratio, even when they try as hard as possible to upload to the network. Whether or not an uploader will be selected by a certain downloader to provide data depends largely on the particular P2P protocol, as well as on the entity that provides a peer with a list of uploaders to choose from (e.g., a centralized tracker).

In Figure 2, we have illustrated various properties of a general P2P file-sharing system and the way in which various actors may influence these properties. Eventually, the relationship between the

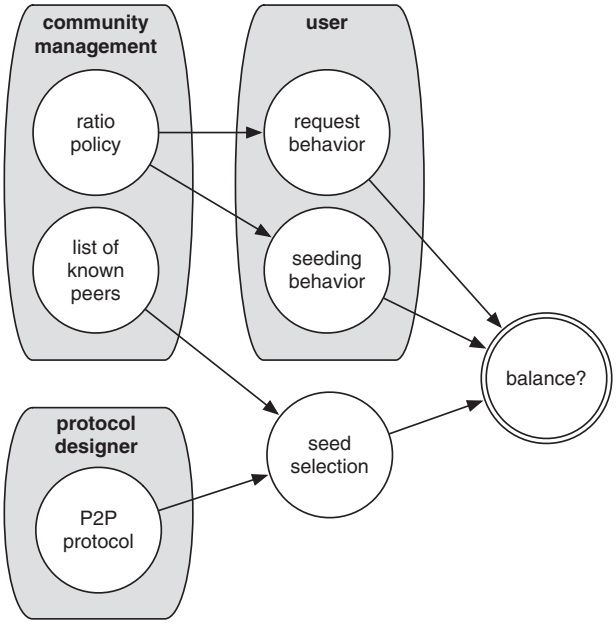


Figure 2. Diagram of the influence of various system properties and their actors on the balance in the system. P2P, peer-to-peer.



DOI: 10.1002/cpe

request behavior, the seeding behavior, and the uploader selection mechanism determine whether a sustainable balance can exist in the system in the long term. If there are many requests while few peers are seeding, the system performance deteriorates, which might eventually cause a collapse of the system. On the other hand, if many peers want to increase their sharing ratios or earn more credits by seeding, there is upload competition.

Even though some communities might explicitly appreciate the banning of peers that do not succeed to upload in spite of their efforts, it is not consistent with the goal of the incentive system, inducing users to seed. This goal is met when users share their files, and therefore, these users should be able to stay in the community. In addition to creating frustration on the part of users (which should be prevented in a healthy community), the effect can even induce a negative loop in which more and more users are being banned, whereas some 'lucky' peers have sharing ratios that become uselessly high.

## 3. MACROECONOMIC SHARING MODEL

We consider a general P2P file-sharing system from a high level, macroeconomic point of view. Next, we describe how we model peers, file requests, the uploader selection algorithm, and the tracker.

### 3.1. Peer model

In real systems, the bandwidths of peers are highly heterogeneous. For instance, Iosup *et al.* [13] have observed in 2005 that 60% of the peers in Europe have a bandwidth in the range 0–100 Kbps, 30% have a bandwidth in the range 100–500 Kbps, and 10% have a bandwidth higher than 500 Kbps. In our heterogeneous scenario, we approximate the bandwidth heterogeneity by assuming that there are $N$ *classes* of peers with upload capacities $u_1, u_2, \ldots, u_N$. We denote the fraction of peers in class $i$ by $\pi_i$.

In line with the goal of this paper, we assume that as a result of an effective SRE mechanism, *each file is seeded by a large number of peers*. In other words, we assume that the supply of bandwidth in the system is not a bottleneck and that the total demand for bandwidth of the downloaders can be satisfied. In our analysis, we also do not consider the download capacity to be a bottleneck. It is not problems due to either a lack of upload or download capacity that we are trying to solve, but we are trying to achieve a more balanced distribution of the upload capacities of the classes across the downloading peers to solve the upload competition problem.

### 3.2. File requests

We consider a large, representative time interval of length $T$, and we assume that during this time interval, $\lambda$ file requests arrive on average per second, for files with an *average* size $s$. In the heterogeneous scenario, we assume that a user's file request behavior is independent of his class, and hence, each request is associated with the various peer classes according to the fractions $\pi_i$. Note that although it matters for the system performance how the file requests are distributed over the interval (e.g., steady state and flashcrowds), this is not relevant from the SRE perspective. It suffices if the interval is long enough for the proportions of originating and filled requests per class to (approximately) match long-run average values—these determine the resulting class sharing ratios.

### 3.3. Uploader selection algorithm

In an *oversupply* scenario in which there are more seeders than downloaders, a downloader has to make a selection of seeders to download from. According to our knowledge, algorithms for this purpose have not received any research attention before. In current BitTorrent implementations, seeders are simply picked at random from the list of available seeders obtained from the tracker. By determining which peers get a chance to upload, the selection algorithm influences the sharing ratios of the different groups of peers and hence which peers 'survive' in the long run. In the next



DOI: 10.1002/cpe

section, we analyze the effect of random selection and show that with this form of selection, the slower peers cannot maintain high sharing ratios.

### 3.4. Tracker

For practical purposes, we assume in our model that there is a central element for peer discovery called a *tracker*. This is common in BitTorrent-like systems. The tracker provides a downloader with a *peer info list*, which contains the IP addresses of a number of other peers in the swarm. The downloader typically contacts these peers to download or exchange data. We note that our main conclusions regarding upload competition also hold for systems with decentralized peer discovery and/or decentralized incentive mechanisms.

## 4. ANALYSIS OF UPLOADER SELECTION

In this section we analyze the currently common random uploader selection algorithm and show that it is not compatible with SRE in a system with heterogeneous bandwidth capacities. Furthermore, we derive a necessary condition that has to hold in a system to prevent a crash of one or more classes.

### 4.1. Random uploader selection

By using the model described in the previous section, we can determine the sharing ratio that each of the *classes* in the system can maintain, depending on their upload capacities. Obviously, the total amount $D_i$ of data downloaded by class $i$ during a time interval of length $T$ is given by

$$D_i = \pi_i \lambda T s. \tag{1}$$

The amount of data uploaded by class $i$ depends on the uploader selection that the downloaders make. The peer info list that a downloader obtains is a random subset of all peers in the system and hence contains a fraction $\pi_i$ of peers of class $i$. A downloader has a certain number of download slots (e.g., limited by the transmission control protocol). When it selects a random set of uploaders from the peer info list, this random set will have a fraction of $\pi_i$ uploaders of class $i$ as well. As peers in class $i$ have upload capacity $u_i$, the number of bytes downloaded from class $i$ by each downloader in the system will be proportional to $\pi_i$ and $u_i$. Hence, the total amount of data $U_i$ that class $i$ has uploaded during the interval of length $T$ is given by

$$U_i = \frac{\pi_i u_i}{\sum_j \pi_j u_j} \lambda T s. \tag{2}$$

As the sharing ratio of a peer is equal to its upload divided by its download, the sharing ratio $R_i$ of class $i$ at time $T$ is given by

$$R_i = \frac{U_i}{D_i} = \frac{u_i}{\sum_j \pi_j u_j}. \tag{3}$$

Although the actual ratio may fluctuate during the interval considered according to the arrival process of peers, the request process of files, and the actual download process, the average sharing ratio in the long run necessarily converges towards the value given by Equation (3).

In Figure 3, we have displayed the average sharing ratio in a system with two classes of peers (slow and fast) in various proportions. The upload capacities of the classes are $u_s$ and $u_f$, respectively. We have computed the average sharing ratio for various differences between the upload capacity of the fast peers and that of the slow peers (the horizontal axis shows the proportion, i.e., $u_f/u_s$). Obviously, when this difference is larger, the difference between their sharing ratios is larger as well. It is striking that in a system with 50% fast peers and 50% slow peers, even if the fast peers are only three times as fast as the slow peers, the average sharing ratio of the slow peers is only 0.5—a ratio which is below the minimum requirement of many communities ([4].) For the scenarios with more fast peers, the sharing ratio obtainable by the slow peers is even lower.



DOI: 10.1002/cpe

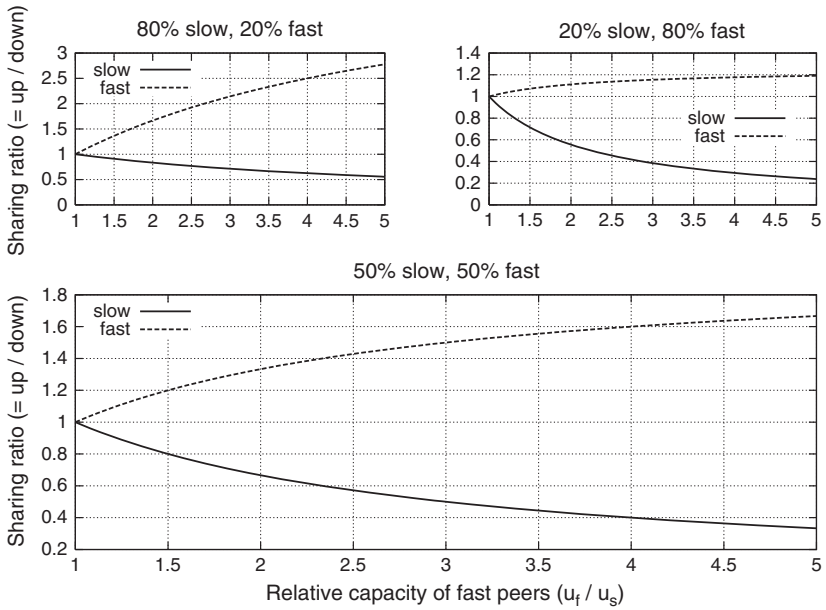


Figure 3. The sharing ratio in various configurations of slow and fast peers.

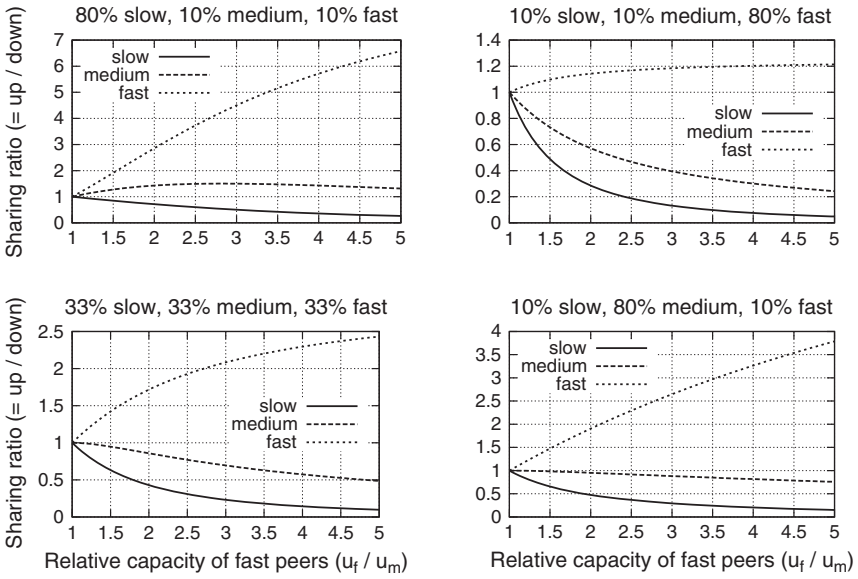


Figure 4. The sharing ratio in various configurations of slow, medium, and fast peers with upload capacities that satisfy $u_m : u_s \equiv u_f : u_m$.

The sharing ratios for a system with three classes (slow, medium, and fast) are displayed in Figure 4. The capacities of the classes are $u_s, u_m$, and $u_f$, respectively. Here, the capacity proportions are chosen such that $u_m : u_s \equiv u_f : u_m$. The average sharing ratio of the fast peers greatly exceeds that of the other two classes in all scenarios. Especially when there are many fast peers, it will not be possible for the slow peers or even the medium peers to maintain a sharing ratio that fulfills the requirements of a common private community.

### *4.2. Unintended banning of all slow peers*

The practical consequence of the aforementioned analysis is the unintended banning of entire peer classes. If a community enforces a sharing ratio of $\delta < 1$ and if $R_i < \delta$ for some class $i$, then at least



DOI: 10.1002/cpe

one peer from this class individually has a sharing ratio below $\delta$. Banning this peer may temporarily improve the class sharing ratio to the extent that $R_i \geqslant \delta$ in the remaining system, but the system will move toward new equilibrium values following from Equation (3). Because the proportion of slow peers in the system has been reduced and therefore the average upload capacity has increased, it is seen from Equation (3) that the sustainable sharing ratio $R_i$ of class $i$ has been reduced. As a consequence, when a new equilibrium is reached, $R_i < \delta$ will certainly hold, resulting in additional banning. This continues until $\pi_i = 0$, that is, until the class has been eliminated completely.

It must be mentioned that ensuring that $R_i \geqslant \delta$ is a necessary but not a sufficient condition to prevent the unfair banning of peers from class $i$, in the sense that it may always happen that individual cooperative peers do not manage to maintain the minimum sharing ratio due to whatever circumstances. Of course, the chance that this happens is smaller when the class as a whole can maintain a sharing ratio well above the minimum.

Apparently, with random uploader selection, the slower classes may not be able to manage to actually provide their fair share of the uploads, a problem that might be remedied by offering them the opportunity to provide more than their fair share.

### *4.3. Generalized random uploader selection*

We now generalize our analysis and assume that with an arbitrary uploader selection algorithm, a fraction $\alpha_i$ of uploaders is selected from class $i$ by every other class. Random uploader selection is a special case, in which $\alpha_i = \pi_i$. Analogous to the derivation of Equation (2), the data uploaded by class $i$ during the interval is given by

$$U_i = \frac{\alpha_i u_i}{\sum_j \alpha_j u_j} \lambda T s. \tag{4}$$

As it still holds that $D_i = \pi_i \lambda T s$, the sharing ratio of class $i$ now becomes

$$R_i = \frac{\alpha_i u_i}{\pi_i \sum_j \alpha_j u_j}. \tag{5}$$

In the design of sustainable SRE, it should at least hold that $R_i \geqslant \delta$. A possible algorithm that satisfies this condition sets $\alpha_i$ such that it implies a sharing ratio $R_i = 1$ for each class $i$. This means that

$$\alpha_i = c \cdot \frac{\pi_i}{u_i} \tag{6}$$

for all $i$, where $c$ is a normalization constant that follows from $\sum_i \alpha_i = 1$. For a two-class system (slow and fast), we find

$$\alpha_s = \frac{\pi_s u_f}{\pi_s u_f + \pi_f u_s}, \quad \alpha_f = \frac{\pi_f u_s}{\pi_f u_s + \pi_s u_f}. \tag{7}$$

### *4.4. Uploader selection in practice*

If the desired values of $\alpha_i$ are known, in theory, an uploader selection algorithm can be designed that simply selects uploaders according to these fractions. However, in practice, such an algorithm is not at all trivial to implement. First of all, a categorization in classes is not always feasible or realistic. Second, the values of $\alpha_i$ might change continuously under dynamic scenarios in which the proportions of the classes change. Third, a downloader changes the peers from which it downloads only periodically. Finally, each downloader makes discrete selections of some number $K$ uploaders from the set of uploaders it knows, which implies a granularity of $1/K$ in the approximation of $\alpha_i$.

In the next section, we present several heuristical algorithms that enable sustainable SRE in a dynamic way.



DOI: 10.1002/cpe

## 5. SUSTAINABLE UPLOADER SELECTION ALGORITHMS

We propose and analyze the following three uploader selection algorithms:

A. Class-based uploader selection
B. Central ratio-based uploader selection
C. Local ratio-based uploader selection

We will analyze these algorithms and describe how they can be implemented in practice.

### *5.1. Class-based uploader selection*

With class-based uploader selection, downloaders receive from the tracker peer info lists consisting of peers from the class that currently has the lowest average sharing ratio. Next, we present the algorithm, and we provide an analysis and experimental results.

*5.1.1. Algorithm.* Algorithm 1 shows the details of class-based uploader selection. In the first piece of code, the tracker first sorts the classes of peers on the basis of their average sharing ratios and then sends peer info lists consisting of a random selection of peers from the class with the lowest average sharing ratio. The second piece of code implements the periodic requests of peers for new peer info lists, and the third piece of code shows that peers pick a random subset of peers from the peer info lists to download from. Note that we assume here that the tracker has divided the peers into classes on the basis of their upload capacities. Of course, in practice, this may be difficult to do, as upload capacities are very diverse, and the peer population changes dynamically. However, this algorithm serves as a theoretical example to provide insight into the generic uploader selection dynamics discussed in the previous section.

*5.1.2. Analysis.* We analyze class-based uploader selection for a system with two classes, slow and fast, and we show that the sharing ratios converge to 1. In Algorithm 1, when the tracker changes

**Algorithm 1** Class-based uploader selection

```
tracker::getPeerList():
  for all class c do
    U_c ← total upload of peers(c)
    D_c ← total download of peers(c)
    R_c ← U_c/D_c
  end for
  R_min ← lowest R_c
  c* ← fastest class c for which R_c = R_min
  peerlist ← sample(peers(c*), N_m)
  return peerlist

peer::updateKnownPeers():
  while true do
    peerlist ← tracker.getPeerList()
    wait τ seconds
  end while

peer::selectUploaders():
  uploaders ← sample(peerlist, N_d)
  for all u in uploaders do
    request data from u (protocol specific)
  end for
```



DOI: 10.1002/cpe

the uploader class selection, it may take up to an amount of time $\tau$ before all peers have received a peer info list with peers of the new uploader class. In the next analysis, we assume that all peers receive such a new peer info list at the same time and that they all start downloading exclusively from peers in the new downloader class immediately.

Let $U_i(t)$ and $D_i(t)$ be the cumulative amounts of data uploaded and downloaded by class $i$ up to time $t$, respectively. The sharing ratio of class $i$ at time $t > 0$ is

$$R_i(t) = \frac{U_i(t)}{D_i(t)}. \tag{8}$$

The preservation of bandwidth in the system implies that $U_s(t) + U_f(t) = D_s(t) + D_f(t)$ for all $t$, which in turn implies that at any time $t$ either $R_s(t)$ and $R_f(t)$ are on the opposite sides of the number 1 or are both equal to 1. We distinguish two states the system can be in:

- `SLOW`: the slow peers are selected for uploading.
- `FAST`: the fast peers are selected for uploading.

The evolution of the sharing ratios is illustrated in Figure 5, where it is assumed that in the time interval considered, the system is initially in state `SLOW`. We shall show that such a situation implies that the sharing ratio of the slow class is increasing and that at some (future) time $t_1 > t$, a crossing of sharing ratios ($R_s(t_1) = R_f(t_1) = 1$) will occur. On the first uploader selection change following $t_1$, say at time $t_1^*$, we will have $R_f(t_1^*) < 1$, and the system switches state to `FAST`. This state is maintained until after the next crossing at $t_2$, the system switches back to state `SLOW` at the first subsequent uploader selection change, and so on. This implies that when the sharing ratios no longer move toward but start moving *away* from the desired equilibrium point, a state change takes place at the earliest possible opportunity, causing the trends to reverse. The largest deviations from the sharing ratio target of 1 therefore occur at the uploader selection change instances, and it suffices to show that the sharing ratios at those instances converge to 1.

We first express the sharing ratio at time $t + \Delta t$ as a weighted average of the sharing ratio at time $t$ and the sharing ratio over the interval $(t, t + \Delta t)$. Consider any class $i$ and write $U = U_i(t)$ and $D = D_i(t)$. By writing $\Delta U = U(t + \Delta t) - U(t)$ and $\Delta D = D(t + \Delta t) - D(t)$ for the amounts accrued over $(t, t + \Delta t)$, we obtain

$$R(t + \Delta t) = \frac{D}{D + \Delta D} \cdot \frac{U}{D} + \frac{\Delta D}{D + \Delta D} \cdot \frac{\Delta U}{\Delta D}, \tag{9}$$

which is indeed a weighted average of $U/D = R(t)$ and $\Delta U/\Delta D$, the sharing ratio over the interval $(t, t + \Delta t)$.

Now, assume that no system state switches occur in the interval $(t, t + \Delta t)$. Then, for the class that is not uploading, $\Delta U/\Delta D = 0$, and so

$$R(t + \Delta t) = \frac{D}{D + \Delta D} \cdot R(t),$$

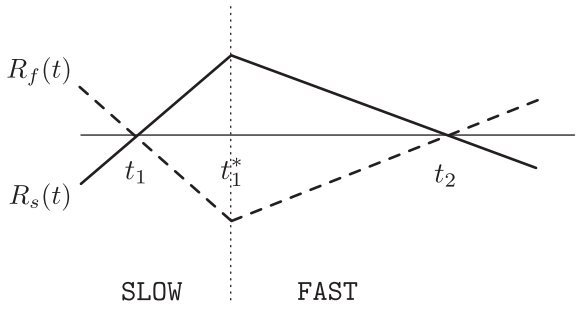


Figure 5. Illustration of the operation of Algorithm 1.



DOI: 10.1002/cpe

which shows that if $\Delta t$ and so $\Delta D$ is large enough, then $R(t + \Delta t) < 1$. For the class that is uploading, say class $i$, we have $\Delta U / \Delta D = 1/\pi_i > 1$: it is responsible for all of the uploading but for only a fraction $\pi_i$ of the downloading. Equation (9) becomes

$$R(t + \Delta t) = \frac{D}{D + \Delta D} \cdot R(t) + \frac{\Delta D}{D + \Delta D} \cdot \frac{1}{\pi_i}, \tag{10}$$

which shows if $\Delta t$ and so $\Delta D$ is large enough, then $R(t + \Delta t) > 1$. This proves that in due course, always a crossing of sharing ratios will occur, as illustrated in Figure 5 at time $t_1$.

Next, let the first uploader selection change after a crossing of sharing ratios at time $t_1$ occur at time $t_1^* = t_1 + \Delta t$; note that $\Delta t < \tau$. Then, Equation (10) can be rewritten as

$$R_i(t_1^*) = 1 + \frac{\Delta D}{D + \Delta D} \left( \frac{1}{\pi_i} - 1 \right). \tag{11}$$

The amount by which $R_i(t_1^*)$ exceeds 1 shrinks as $t$ goes to infinity because $\Delta D$ is bounded and $D = D(t)$ goes to infinity. A similar reasoning applies to the downward deviation from 1 of the sharing ratio of the class that is not uploading. We conclude that the oscillations of the sharing ratios gradually become smaller, which proves the convergence claim.

This analysis can straightforwardly be extended to systems with arbitrary numbers of classes.

*5.1.3. Experimental results.* By using a scenario with 50% slow peers, and 50% fast peers, we have computed the evolution of the sharing ratio in both classes when class-based uploader selection is applied. We have assumed that downloaders update their selection every 10 min, although in practice, this can be set to any desired value. In Figure 6, the sharing ratio for both classes is displayed. After some period of stabilization, we clearly see that the ratio for both classes converges to 1. We also perform this experiment for different proportions of fast/slow peers, which show the same pattern of stabilization. In Figure 7, we have displayed the average fraction of time that the slow peers are selected for uploading. The value to which this fraction converges, 0.8, is exactly the value which follows from Equation (6).

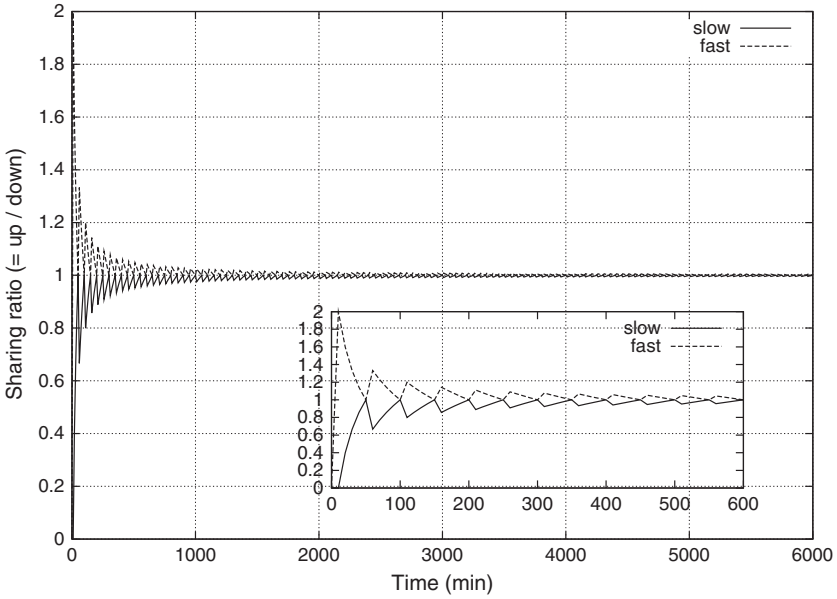


Figure 6. The sharing ratio of slow and fast peers under class-based uploader selection where $u_f : u_s = 4 : 1$ and $\tau = 10$ min.



DOI: 10.1002/cpe

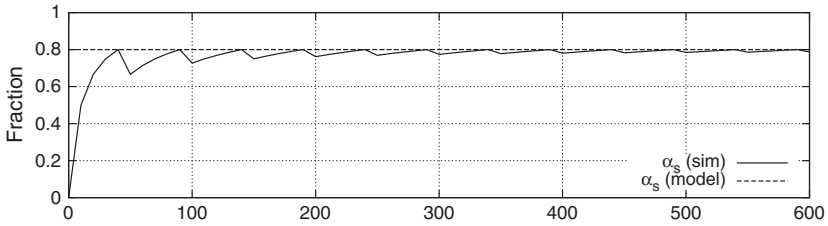


Figure 7. The fraction of time the slow peers are selected for uploading ($\alpha_s$) under the class-based uploader selection of Figure 6.

### 5.2. *Central ratio-based uploader selection*

With central ratio-based uploader selection, downloaders receive from the tracker peer info lists consisting of the peers that currently have the lowest sharing ratios. Next, we present the algorithm and we provide some implementation considerations and experimental results.

*5.2.1. Algorithm.* Algorithm 2 shows the code of the central ratio-based uploader selection algorithm. In the first piece of code, the tracker first sorts the peers according to their current sharing ratios and then sends peer info lists consisting of the peers with the lowest sharing ratios. The pieces of code executed by the peers is the same with class-based uploader selection. In particular, a downloading peer has no knowledge of the sharing ratios of the peers it can select to download from.

*5.2.2. Implementation considerations.* As all essential operations of Algorithm 2 are performed by the tracker, the implementation is relatively straightforward. Moreover, this solution is attractive because, as it does not require changing the client, it is completely compatible with existing BitTorrent clients and similar clients. Community designers could simply modify the code of their tracker to influence the selection of uploaders. A practical remark, however, is that current clients do not discard the peers that they already know when they receive a new list of peers from the tracker.

**Algorithm 2** Central ratio-based uploader selection

```
tracker::getPeerList():
  for all peers p do
    R_p ← U_p/D_p
  end for
  sortedlist ← sort peers with respect to R_p (ascending)
  peerlist ← sortedlist[1 : N_m]
  return peerlist

peer::updateKnownPeers():
  while true do
    peerlist ← tracker.getPeerList()
    wait τ seconds
  end while

peer::selectUploaders():
  uploaders ← sample(peerlist, N_d)
  for all u in uploaders do
    request data from u (protocol specific)
  end for
```



DOI: 10.1002/cpe

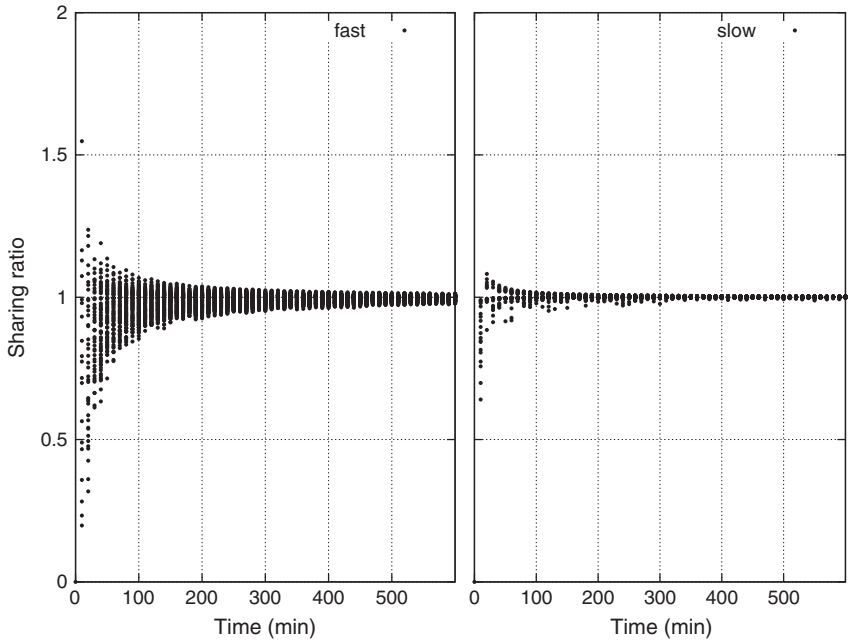


Figure 8. Scatter plot of the sharing ratios of fast peers (left) and slow peers (right) over time under central ratio-based uploader selection with $N_m = 20$.

A known peer is only discarded when it is no longer available or connectable. Hence, it is likely that the effect of central ratio-based uploader selection is somewhat delayed when implemented only at the tracker.

A second important consideration is the setting of the parameter $N_m$, the number of peers that the tracker sends to the downloaders (i.e., the peer info list size). Because downloaders make a random selection of uploaders from the peers they know, the larger the list of known peers, the less influential the bias towards peers with low sharing ratios. In dynamic systems, larger lists of peers generally benefit the robustness of the download process. Hence, a practical trade-off is unavoidable.

Finally, to compute the sharing ratio of a peer, the tracker has to know its amounts of uploaded and downloaded data. In real private trackers on the Internet, this information is provided by the peers themselves[¶]. A variety of mechanisms have been suggested that aim to obtain such information in a secure way (see, for example, [14]). However, as remarked before, we consider a scenario in which a *successful* SRE mechanism is already in place, regardless of its technical details.

*5.2.3. Experimental results.* We are interested in the overall, long-term dynamics of central ratio-based selection, in particular with respect to the influence of the size $N_m$ of the peer info list provided by the tracker. We have simulated several scenarios, each with 500 slow peers and 500 fast peers. The upload capacity of the fast peers is four times as large as that of the slow peers. We do not consider individual differences in request behavior or the semantics of what is downloaded. Each peer is periodically performing a 'download' of some amount of (undefined) data at discrete time intervals from a selection of $N_d = 5$ uploaders consistent with Algorithm 2. For each request, we assume that a random selection of 100 peers in the system are possible candidates for uploading (which roughly emulates the notion of a swarm). Every $\tau = 10$ min, a peer requests a new peer info list from the tracker. To test the 'self-stabilization' of the algorithm, all peers start with a sharing ratio of 0.

In Figures 8–10, we have displayed scatter plots of the sharing ratios of the peers for $N_m = 20$, $N_m = 50$, and $N_m = 100$, respectively. Clearly, when $N_m = 20$, the effect of the algorithm is very

[¶]Private trackers usually have a primitive security mechanism in place; for each transaction, the information provided by the sending peer is compared with the information provided by the receiving peer. Peers that are repeatedly suspected of providing false information are banned from the tracker.



DOI: 10.1002/cpe

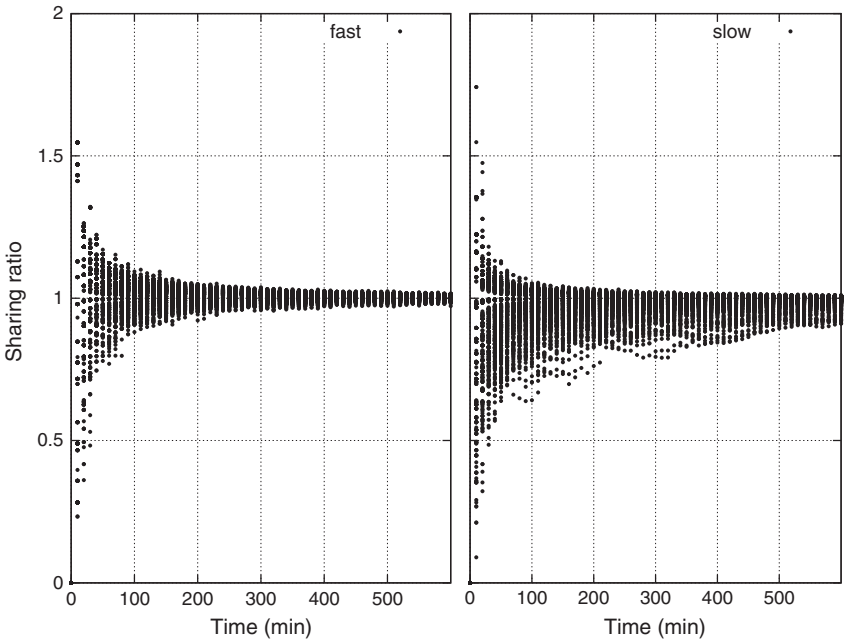


Figure 9. Scatter plot of the sharing ratios of fast peers (left) and slow peers (right) over time under central ratio-based uploader selection with $N_m = 50$.

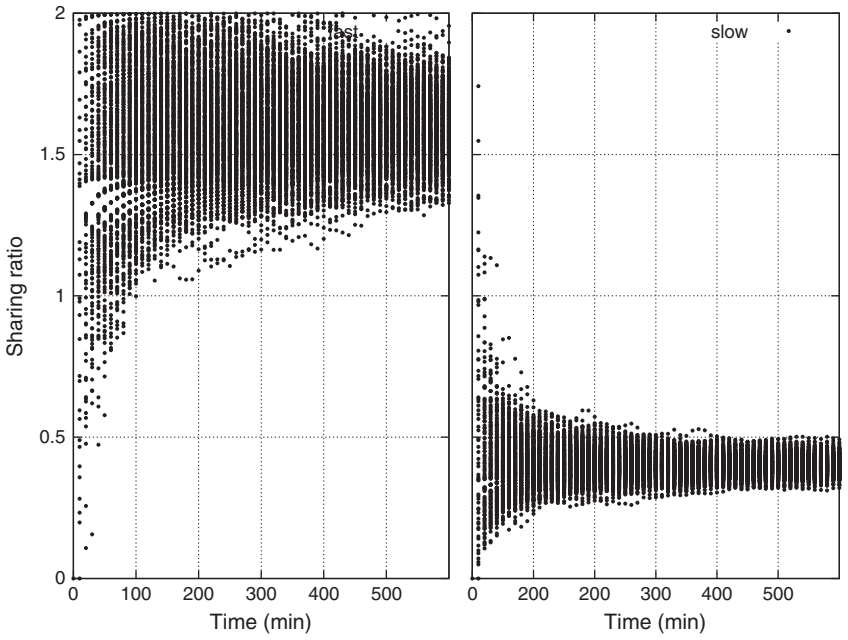


Figure 10. Scatter plot of the sharing ratios of fast peers (left) and slow peers (right) over time under central ratio-based uploader selection with $N_m = 100$.

strong, and the sharing ratios of all peers quickly converge to 1. The deviation among the fast peers is slightly larger than the deviation among the slow peers; this is intuitive because for a fast peer, the impact on its sharing ratio during an interval in which it is selected is larger than for a slow peer. At $N_m = 100$ on the other extreme, the effect of the algorithm has completely vanished, which is obvious because the total size of the 'swarm' is 100, so selecting 100 peers from it introduces no bias. The really interesting observation is how effective the algorithm still is with $N_m = 50$.



DOI: 10.1002/cpe

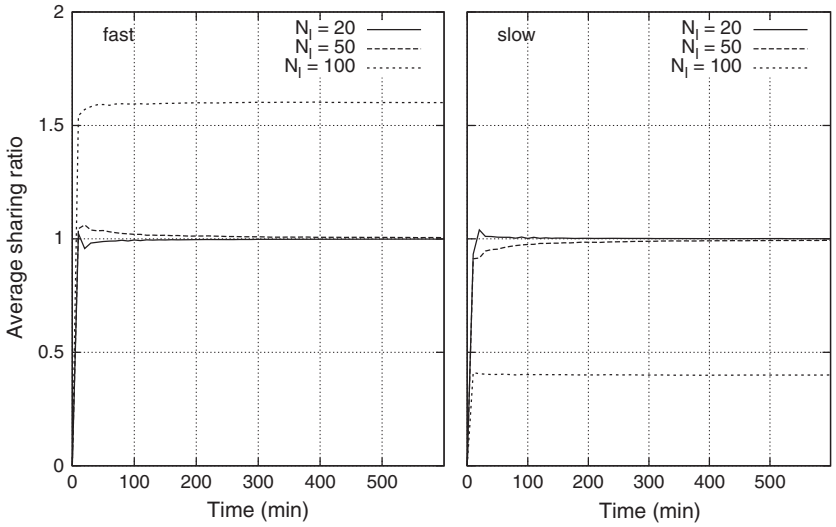


Figure 11. The average sharing ratio of fast peers (left) and slow peers (right) under central ratio-based uploader selection for $N_m = 20, 50$, and 100.

**Algorithm 3** Local ratio-based uploader selection

```
tracker::getPeerList():
  for all peers p do
    R_p ← U_p/D_p
  end for
  ratiolist ← {(p, R_p) : p in peers}
  peerlist ← sample(ratiolist, N_m)
  return peerlist

peer::updateKnownPeers():
  while true do
    peerlist ← tracker.getPeerList()
    sortedlist ← peerlist sorted with respect to R_p (ascending)
    wait τ seconds
  end while

peer::selectUploaders():
  uploaders ← sortedlist[1 : N_d]
  for all u in uploaders do
    request data from u (protocol specific)
  end for
```

Clearly, the difference with $N_m = 100$ is very significant and the converging effect of the algorithm is clearly visible, despite some deviation among the slow peers. In Figure 11, we have plotted the average sharing ratios for all three scenarios. The impact of the value $N_m$ is even more striking here. Apparently, $N_m = 50$ is 'small' enough to have a very strong effect because the average sharing ratio of all peers is very close to 1. However, with $N_m = 100$, the difference between the sharing ratio of the fast peers and that of the slow peers is very large; note that the values are exactly as predicted in our analysis of random uploader selection in Figure 3 for $u_f/u_s = 4$.



DOI: 10.1002/cpe

### 5.3. Local ratio-based uploader selection

With local ratio-based uploader selection, downloaders receive from the tracker peer info lists consisting of random peers, and it is the downloaders themselves that select uploaders with the lowest sharing ratios from among the peers they know. Next, we present the algorithm, and we provide some implementation considerations and experimental results.

*5.3.1. Algorithm.* In Algorithm 3, the details of the algorithm are displayed. In the first piece of code, the tracker sends peer info lists to downloaders consisting of random peers along with their sharing ratios. In the second piece of code, peers sort the peers they know according to their sharing ratios, and in the third piece of code, they select the ones with the lowest sharing ratios to download from.

*5.3.2. Implementation Considerations.* With this algorithm, both the code at the tracker and the code at the client would have to be adapted. The tracker can easily compute the sharing ratios of peers, and it is relatively straightforward to extend the peer info message with a single field. Also, sorting at the side of the client is simple to implement. However, this approach clearly needs more effort to implement than the central algorithm, and it is not backward compatible with existing clients. It is therefore more suitable for newly designed systems and protocols.

Furthermore, if the tracker would not have sharing-ratio information available, this information might be collected by the peers themselves in a decentralized way, such as with a decentralized reputation system [1].

*5.3.3. Experimental results.* We performed experiments for local ratio-based uploader selection in the same way as for central ratio-based selection discussed earlier. In Figures 12 and 13, we have displayed scatter plots for peer info lists with sizes $N_m = 20$ and $N_m = 100$, respectively. The plots show that the local sorting approach is effective for both scenarios; it is obvious that in both cases, the sharing ratio converges quickly to 1. As the candidates for sorting are fewer in the case of $N_m = 20$, the variance in the sharing ratios is slightly higher than with $N_m = 100$. It is intuitive that

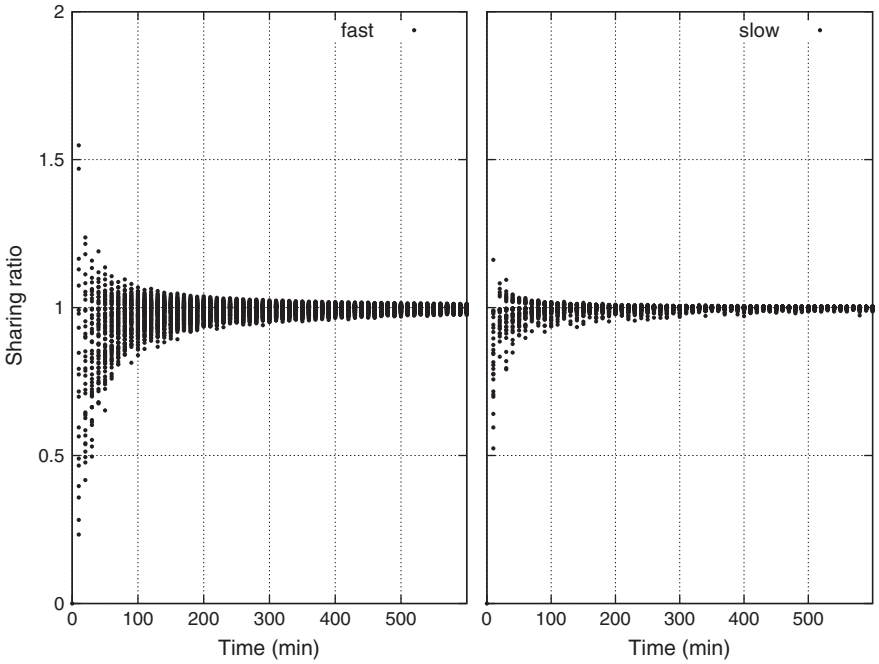


Figure 12. Scatter plot of the sharing ratios of fast peers (left) and slow peers (right) over time under local ratio-based uploader selection with $N_m = 20$.



DOI: 10.1002/cpe

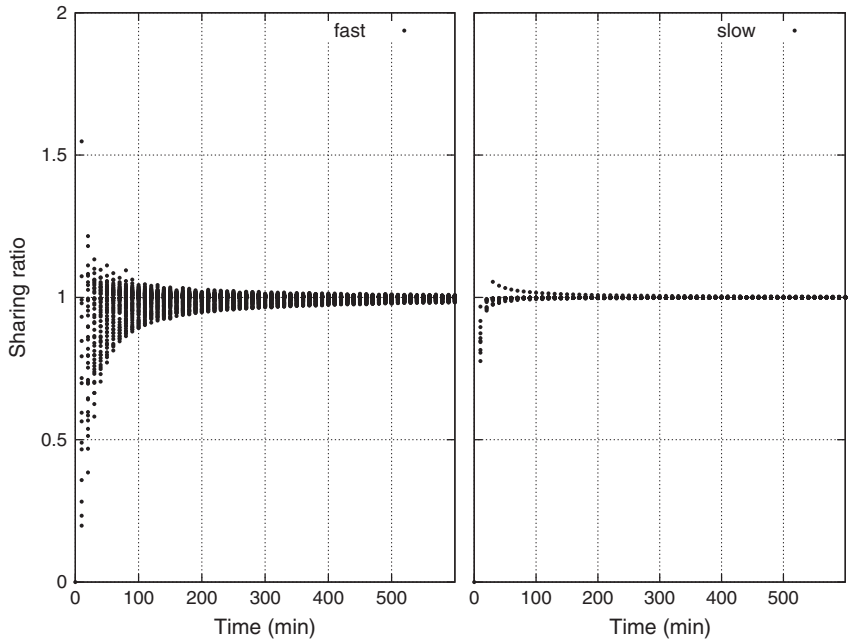


Figure 13. Scatter plot of the sharing ratios of fast peers (left) and slow peers (right) over time under local ratio-based uploader selection with $N_m = 100$.

a larger selection to choose from yields better results. Note that this correlation is 'opposite' from the correlation in the central case, which yields better results for shorter info lists . Overall, we can conclude that local selection is more effective than central selection but less attractive to implement. In practice, it is up to system designers or community managers to make this trade-off, depending on their needs.

## 6. SECURITY

In incentive mechanism design, the question of security is central. As the common assumption that peers tend to free ride and to minimize their contributions, an incentive mechanism should be resistant against certain forms of cheating and manipulation. As we have demonstrated, private communities are able to induce cooperation in practice, whereas because of their central administration, cheating is generally prevented (although it is possible on some trackers).

The problem that we focus on in this section has a very different angle. We are not concerned with the question whether a peer is willing to seed—the underlying incentive mechanism is responsible for that. Rather, we are concerned with whether there is an incentive for a downloader to change its *local* uploader selection to improve its download performance, thereby possibly subverting the sharing-ratio mechanism. With class-based uploader selection, there is obviously no potential gain to be obtained from doing so. With central ratio-based uploader selection, the potential gain is not likely to be significant as the tracker limits the selection of uploaders by downloaders to the peers that need the chance to upload most. With local ratio-based uploader selection however, peers do have the opportunity to pick fast peers rather than peers with low sharing ratios as uploaders. If many peers would thus subvert the protocol, the ratio balancing might be undermined. This possibility is inevitable when the decision as to which uploaders to download from is completely local. Hence, such a local mechanism might only be suitable for communities in which not many peers would actually subvert the protocol or where certain additional mechanisms are deployed, which detect cheaters.

Overall, as each of the presented mechanisms has different advantages and disadvantages, it is up to community managers to make the necessary trade-offs to select the mechanism that best suits their needs.



DOI: 10.1002/cpe

## 7. RELATED WORK

To tackle the issue of free riding, or cooperation in general, many incentive mechanisms have been proposed over the last decade. However, we are not aware of any work that explores *upload competition* or *uploader selection* in P2P systems. Although there are several papers on seeding strategies, such as by Esposito *et al.* [15], we note explicitly that these strategies determine how a *seeder* makes a selection among *many downloaders* during heavy demand, which is the inverse of the scenario we focus on. Nevertheless, we will present a selection of more general work in the field of incentive mechanisms that served as a background to our work.

In our own earlier work [1], we present *BarterCast*, a reputation-based incentive mechanism that is based on a gossip-like information distribution protocol. Piatek *et al.* [2] present a mechanism for one-hop reputation, which can be used to provide reputation-based incentives. Furthermore, Nandi *et al.* [9] present *Scrivener*, a fully decentralized system to ensure fair sharing, and Landa *et al.* [8] present a sybilproof indirect reciprocity mechanism. Liu *et al.* [16] propose an upload entropy scheme that limits colluding, while rarely affecting normal users. Also, a variety of credit mechanisms has been presented. Vishnumurthy *et al.* [3] present *Karma*, a secure economic framework for P2P resource sharing; Garcia *et al.* [10] present a decentralized adaptation of this same framework. Moreton *et al.* [11] present a generic analysis of token-based accounting mechanisms, whereas Hausheer *et al.* [12] relate such accounting mechanisms to market mechanisms in P2P systems.

A variety of measurements of BitTorrent communities have been presented in other work. In our own earlier work [4], we present the results of measurements of over half a million peers in five public and private communities. Zhang *et al.* [5] provide a broad overview of the private BitTorrent landscape consisting of more than 800 private sites, which they call ‘darknets’. Andrade *et al.* [6] analyze supply and demand in a selection of public and private communities.

Finally, we remark that our work is fully compatible with several proposals of locality-aware neighbor selection (e.g., [17]).

## 8. CONCLUSIONS

In this paper, we have investigated the problem of *upload competition* that may occur in P2P systems with successful incentive systems when peers are competing for chances to upload to maintain sufficiently high sharing ratios. By using a macroeconomic sharing ratio model, we have demonstrated that current naive random uploader selection algorithms are not suitable for sustaining a balance in such systems because if all peers are allowed to upload for the same fraction of time, peers with relatively low upload capacities will inevitably have low sharing ratios despite their willingness to share, thus running the risk of being banned from the system by the sharing ration enforcement mechanism. We have proposed and analyzed several sustainable uploader selection algorithms that are effective and easy to deploy in real systems. Our central ratio-based upload selection mechanism is based on a preselection of peers with low sharing ratios at the tracker, requiring only a small change in the tracker code while being highly effective in balancing the ratios in the system. Our local ratio-based upload selection mechanism is based on local selection of uploaders at the downloading peers, which requires changes in the client code as well. The latter algorithm can also be adapted to a fully distributed incentive system, such as the BarterCast protocol [1]. Overall, we are convinced that uploader selection mechanisms are necessary to sustain successful P2P communities on the long term.

## REFERENCES

1. Meulpolder M, Pouwelse J, Epema D, Sips H. Bartercast: A practical approach to prevent lazy freeriding in p2p networks. *Proceedings of the 23rd IEEE International Symposium on Parallel and Distributed Processing (IPDPS’09)*, Rome, Italy, 2009.
2. Piatek M, Isdal T, Krishnamurthy A, Anderson T. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI’08)*, San Francisco, CA, USA, 2008.



DOI: 10.1002/cpe

3. Vishnumurthy V, Chandrakumar S, Sirer E. Karma: a secure economic framework for peer-to-peer resource sharing. In *Proceedings of the 2nd Workshop on Economics of Peer-to-Peer Systems (P2PECON'03)*, Cambridge, MA, USA, 2003.
4. Meulpolder M, D'Acunto L, Capotă M, Wojciechowski M, Pouwelse J, Epema D, Sips H. Public and private bittorrent communities: a measurement study. In *Proceedings of the 9th International Workshop on Peer-to-Peer Systems (IPTPS'10)*, San Jose, CA, USA, 2010.
5. Zhang C, Dhungel P, Liu Z, Ross K. Bittorrent darknets. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM'10)*, San Diego, CA, USA, 2010.
6. Andrade N, Santosneto E, Brasileiro F, Ripeanu M. Resource demand and supply in BitTorrent content-sharing communities. In *Computer Networks* November 2008; **53**(4):515–527.
7. Adar E, Huberman BA. Free riding on Gnutella. *Technical report*, Palo Alto, CA, USA, August 2000.
8. Landa R, Griffin D, Clegg R, Mykoniati E, Rio M. A sybilproof indirect reciprocity mechanism for peer-to-peer networks. In *Proceedings of the 28th Conference on Computer Communications (INFOCOM'09)*, Rio de Janeiro, Brazil, 2009.
9. Nandi A, Ngan TW, Singh A, Druschel P, Wallach DS. Scrivener: providing incentives in cooperative content distribution systems. In *Proceedings of the 6th International Middleware Conference*, Grenoble, France, 2005.
10. Garcia FD, Hoepman JH. Off-line karma: a decentralized currency for peer-to-peer and grid applications. In *Proceedings of the 3rd International Conference on Applied Cryptography and Network Security (ACNS'05)*, New York, NY, USA, 2005.
11. Moreton T, Twigg A. Trading in trust, tokens, and stamps. In *Proceedings of the 2nd Workshop on Economics of Peer-to-Peer Systems (P2PECON'03)*, Cambridge, MA, USA, 2003.
12. Hausheer D, Liebau N, Mauthe A, Steinmetz R, Stiller B. Token-based accounting and distributed pricing to introduce market mechanisms in a peer-to-peer file sharing scenario. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P'03)*, 2003.
13. Iosup A, Garbacki P, Pouwelse JA, Epema DHJ. Correlating topology and path characteristics of overlay networks and the internet. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, Singapore, 2006.
14. Feldman M, Lai K, Stoica I, Chuang J. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*, New York, NY, USA, 2004.
15. Esposito F, Matta I, Michiardi P, Mitsutake N, Carra D. Seed scheduling for peer-to-peer networks. In *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (NCA'09)*, Cambridge, MA, USA, 2009.
16. Liu Z, Dhungel P, Wu D, Zhang C, Ross KW. Understanding and improving ratio incentives in private communities. In *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS'10)*, Genoa, Italy, 2010.
17. Bindal R, Cao P, Chan W, Medved J, Suwala G, Bates T, Zhang A. Improving traffic locality in bittorrent via biased neighbor selection. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS'06)*, Lisboa, Portugal, 2006.



DOI: 10.1002/cpe