2Fast : Collaborative downloads in P2P networks

Paweł Garbacki, Alexandru Iosup, and Dick Epema
Delft University of Technology
{*p.j.garbacki,a.iosup,d.h.j.epema*}*@tudelft.nl*

Maarten van Steen
Vrije Universiteit Amsterdam
steen@cs.vu.nl

Abstract

P2P *systems that rely on the voluntary contribution of bandwidth by the individual peers may suffer from freeriding. To address this problem, mechanisms enforcing fairness in bandwidth sharing have been designed, usually by limiting the download bandwidth to the available upload bandwidth. As in real environments the latter is much smaller than the former, these mechanisms severely affect the download performance of most peers. In this paper we propose a system called 2Fast, which solves this problem while preserving the fairness of bandwidth sharing. In 2Fast, we form groups of peers that collaborate in downloading a file on behalf of a single group member, which can thus use its full download bandwidth. A peer in our system can use its currently idle bandwidth to help other peers in their ongoing downloads, and get in return help during its own downloads. We assess the performance of 2Fast analytically and experimentally, the latter in both real and simulated environments. We find that in realistic bandwidth limit settings, 2Fast improves the download speed by up to a factor of* 3.5 *in comparison to state-of-the-art* P2P *download protocols.*

1. Introduction

In the P2P research community the role of content transfer is often neglected in favor of content searching. Experience with widely used P2P applications such as file sharing networks shows, however, that the time needed to locate the data is only a small fraction of the time required to actually transfer the data to the requester. State-of-the-art P2P data transfer protocols such as BitTorrent [3] have decreased the time needed to fetch a medium-sized file from days to hours. In this paper we show how this time can be further reduced to minutes.

Existing P2P download protocol designs take one of the following two approaches. They either assume that users are willing to contribute resources even without being explicitly rewarded by the system for their contributions, or they employ mechanisms enforcing fair resource sharing.

The assumption that there is enough altruism in the system limits the applicability of a solution to communities of trusted users. However, Internet-scale deployment where users are anonymous may lead to protocol abuses by freeriders [1], i.e., users obtaining data without contributing, which results in a drastic performance drop. In systems like Gnutella [12] and Kazaa [6], users have no natural incentive to provide services to other participants. In this respect the users of P2P systems closely resemble rational agents [14] who are willing to follow the protocol only if their behavior maximizes their utility. If there is no immediate reward for cooperative behavior, nodes will behave selfishly, which will bring down the performance of the whole system, a phenomenon known as the "tragedy of commons" [5].

The performance decrease of P2P systems caused by freeriders has stimulated research on mechanisms to enforce fair resource sharing [7, 14]. Practice shows, however, that the constraints conditioning a user's quality of service on his actual contribution to the community are too restrictive in many cases. The tit-for-tat bartering protocol of BitTorrent [3], for example, limits peer's download bandwidth by the upload link capacity, which prevents users connected with ADSL links from utilizing their full download bandwidth. Keeping in mind that the majority of P2P users are connected through such asymmetric links [13], this restriction affects the performance of the whole system.

In this paper we describe, analyze and evaluate a P2P protocol called 2Fast that prevents freeriding, but at the same time allows forming collaborative groups of peers that help each other in downloading files. The bandwidth resources are shared among peers inside such groups, thus improving their download performance. The bandwidth sharing model of 2Fast is significantly different from the models of existing P2P data transfer systems. While alternative approaches based on the fair bandwidth sharing paradigm allow for interaction only between certain peers, e.g., peers downloading the same file, in 2Fast any peer with idle bandwidth can participate in a collaborative download. In 2Fast, the mechanisms enforcing fairness between collaborating and non-collaborating peers are separated from each other, allowing collaborations based on different fairness models to coexist in a single system.

Collaborations in 2Fast are embedded in an environment using tit-for-tat [3] data exchange between peers. The tit-for-tat mechanism provides a simple means of enforcing fairness between anonymous peers, e.g., peers from outside a single collaboration. Because of the asymmetric nature of the collaboration relation, tit-for-tat is not applicable inside a collaboration. Instead, we suggest exploiting social

phenomena such as friendships between users as an incentive to form collaborations. In this respect 2Fast follows the social-based P2P system design paradigm introduced in our earlier work [10]. That work has given rise to the P2P client called Tribler. The fully functional implementation of 2Fast has been integrated with the Tribler code base and is publicly available for download at the project page — `http:\\tribler.org`.

2. Implications of fairness on P2P data transfer

For many researchers the role of the P2P overlay is limited to locating files. In reality, however, also the efficient fetching of content, once it has been located, is a nontrivial problem, particularly if fairness in bandwidth contributions is one of the system design objectives.

The data transfer protocols employed by the early P2P systems such as Gnutella [12], and Kazaa [6] lack of explicit measures of fairness in bandwidth usage which makes them an easy target for freeriders [1, 15]. It was shown in [1] that 70% of Gnutella users exploit the generosity of others by downloading shared files without contributing anything to the common good.

BitTorrent [3] is the first widely used P2P file download protocol that incorporates fairness enforcement mechanisms. Every BitTorrent node that has acquired some subset of the file trades blocks with other peers until it has the whole file. In order to bootstrap new nodes, peers reserve a fraction (usually $1/4$) of their bandwidth for altruistic service. Nodes that trade fairly their bandwidth usually experience a higher quality of service.

A serious limitation of the BitTorrent-derived systems is that they cannot preserve information about peer contributions in between download sessions. Consequently, peers can only use the service with the quality of their current contribution. In terms of bandwidth, this statement translates to the peers' effective download bandwidth being limited by their upload link capacity. In case of upload-bandwidth-limited peers, such as peers connected through asymmetric links, this constraint seriously downgrades the achievable download performance.

Our 2Fast system extends the bartering model of BitTorrent-like file transfer protocols. The basic concept of 2Fast is that a peer that has idle bandwidth to spare can join a download currently in progress, fetching missing fragments of the downloaded file and uploading them to the download initiator. The contributed bandwidth can be reclaimed in the future when the peer needs additional bandwidth to speed up its own download.

3. The design of 2Fast

In this section we describe the model of the environment in which 2Fast is deployed, we present the design of the 2Fast system, and we discuss the mechanisms providing incentives for formation of collaborations of peers in 2Fast system.

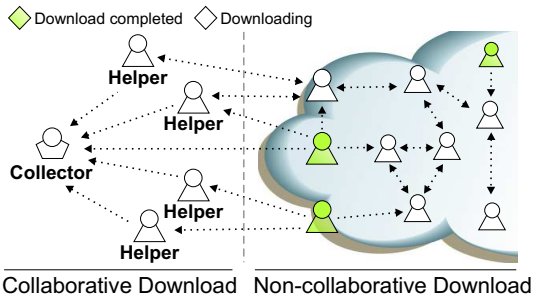

Figure 1. Collector and helpers.

3.1. The data distribution model

The 2Fast protocol targets environments where large files are downloaded by many users at the same time. An example of such an environment is the BitTorrent network. BitTorrent is currently the largest P2P file-sharing network with over one third of the world's P2P traffic [8], which constitutes more than 15% of the total Internet traffc.

In our approach we focus on optimizing data transfer speed rather than search capabilities leaving the data location problem aside. We assume that data exist in the network in the form of files or file archives (i.e., files containing multiple files). To facilitate the file exchange process, files are split in smaller parts called *chunks*. The size of a chunk in a BitTorrent is in the range of a few hundred kilobytes. While selecting chunks to download from each other, peers follow the *rarest-first* policy [3], meaning that the chunk with the fewest replicas is selected for download first. A peer has a complete file only after obtaining all the chunks composing that file. A peer that has the whole copy of a file is called a *seeder*, while a peer whose download is still in progress is referred to as a *leecher*. The group of peers (seeders and leechers) downloading a particular file is referred to as a *swarm*.

3.2. The concept of collaborative downloads

In the 2Fast protocol, a peer takes one of two roles: it is either a *collector* or a *helper* (see Figure 1). The collector is the peer that is interested in obtaining the complete copy of a particular file. In order to improve the performance of the download, a collector forms a collaboration consisting of peers that agree to become its helpers. The helpers download distinct chunks of the file from the peers outside the collaboration they are in, and send these chunks to the collector without requesting any other chunk in return. The collector optimizes its download performance by dynamically selecting the best available data source from the set of helpers and other peers in the network. Helpers give priority to requests originating from the collector and are therefore preferred as data sources. Helpers are not necessarily interested in the content they are downloading.

The fairness of bandwidth sharing is enforced by a hybrid approach, as the bandwidth is accounted differently between non-collaborating peers, and between a collector and its helpers. Peers that are not members of the same collaboration can exchange chunks with the standard BitTorrent tit-for-tat [3] algorithm. The BitTorrent tit-for-tat bartering strategy ensures that for a peer the amount of incoming data is roughly equal to the amount of outgoing data.

The tit-for-tat mechanism guarantees fairness only within a single download session (download of one file). The asymmetric nature of the collector-helper relation implies that the bandwidth contributed by the helper to obtain the chunks for the collector in the current download can be reclaimed only during later downloads. We base the fairness between collaborating peers on a notion of a *promise*. A collector recruits helpers by giving them a promise that the bandwidth invested in boosting the collector's download performance will be returned in the future. Helpers contribute their currently idle bandwidth knowing that later the roles may be reversed — a helper will become a collector and the current collector will pay back for the consumed bandwidth by participating in the download as a helper.

The role of a promise is to provide an incentive for collaboration. This role can be fulfilled only if the system provides guarantees that the promise will be delivered. The 2Fast system in its basic form uses *social incentives* to enforce promise delivery. In our previous work [10] we have shown that social phenomena such as friendship and the existence of communities of users who trust each other can be exploited in P2P protocol design. Viewing users as social partners rather than solitary rational agents [14] can alleviate the problem of false promises, by exploiting the fact that people tend not to steal bandwidth from the social group they belong to. The observation that members of social communities tend to have significant numbers of friends [10] in combination with the small number of helpers that is required to achieve high download performance (see Section 4) gives a reason to believe that a collector should not have problems in finding helpers. In the system described in [10], users select who they are willing to help by explicitly specifying how much they trust each other. The trust value can be modified at any time to reflect the change in users' confidence in each others' credibility, e.g., as a consequence of (not) keeping promises. This solution is similar to the inter-tribal exchanges of cowrie (shells) for seasonal food products, used from as early as 1200 BC to the mid 20th century in bankless transactions [4].

3.3. Bandwidth sharing model of 2Fast

The 2Fast protocol enables a new, less restrictive model of resource sharing in P2P data distribution networks. The existing P2P data distribution systems base their incentives on *content trading*, while 2Fast introduces *bandwidth trading* based incentives.

All content trading incentives translate, in one way or another, to gaining the right to download a file from the network in return for uploading a file stored locally to the network. In this model, files or content in general, are traded between peers to enforce the fairness of sharing. Enabling content trading between two peers in the simplest situation requires that each of the involved peers has content which is interesting for the other one. As a consequence, content trading can happen only between peers that have mutual content interests. The BitTorrent protocol is an example of the most restrictive variant of content trading as BitTorrent peers trade chunks of the same file. Systems such as Scrivener [7] adopt a more advanced content trading mechanism called *transitive trade*. Transitive trade establishes a *credit path* [7, 2] from the requesting node to the node that has the desired file. Credits are then transferred along this path and the download may start. Discovering credit paths is, however, a complex problem, and there is always a chance that no path exists between two particular peers.

In the bandwidth trading incentive model of 2Fast the traded good is bandwidth rather than content. Any peer can act as a helper to any other peer, investing its currently underutilized bandwidth and getting this bandwidth returned in the future. No mutual interest between the helper and the collector is therefore required.

4. Performance analysis of 2Fast

In this section we analyze the performance of the 2Fast collaborative download protocol.

4.1. Notation

For the purpose of the analysis we introduce the following notation:

N the number of leechers in the system,

S the number of seeders in the system,

K the number of chunks that the file distributed in the swarm is divided into,

L the number of peers that possess a particular chunk. The rarest-first chunk selection policy guarantees that every chunk has roughly the same number of copies in the network,

n_i the number of chunks possessed by peer i,

$m_{i,k}$ 1 if peer i possesses chunk k, 0 otherwise,

μ the maximal upload bandwidth of a peer. For the purpose of the formal analysis we assume that all peers have the same maximal upload bandwidth,

c the maximal download bandwidth of a peer. We assume that all peers have the same maximal download bandwidth, and that $c \geq \mu$.

4.2. The selection of the number of helpers

The performance of the collaborative download depends on the number of helpers that are involved. It is clear, however, that after the collector's bandwidth is filled, adding a new helper won't improve the download speed. Here, we compute the minimal number of helpers sufficient to fill the collector's download bandwidth.

The *effective download bandwidth* of the collector is the sum of three contributions: the bandwidth obtained from the seeders, the bandwidth bartered for with peers outside the collaboration, and the bandwidth provided by the helpers. As to the first contribution, the aggregate upload bandwidth of the seeders in the system is $S\mu$. So, assuming that this bandwidth is distributed equally over the N leechers, each peer (and so, also the collector) gets from the seeders a constant download rate of $S\mu/N$. Secondly, the tit-for-tat mechanism of BitTorrent guarantees that the collector's download bandwidth from the peers it is bartering with roughly equals its upload bandwidth μ. Thirdly, assuming that the collector has h helpers, let $f_i \in (0, 1]$ be the fraction of the upload bandwidth of helper i which is used to transfer data to the collector, for $i = 1, 2, \ldots, h$. We can now express the effective download bandwidth d of the collector as

$$d = \frac{S}{N} \cdot \mu + \mu + \sum_{i=1}^{h} f_i \mu. \tag{1}$$

In this paper we assume that $c > S\mu/N + \mu$, for otherwise the maximal download bandwidth of the collector is already filled by the seeders and by bartering, and helpers are of no use.

Because a fraction f_i of the upload bandwidth of helper i is reserved for the collector, only a fraction of $1 - f_i$ can be spent for bartering with other peers, which results in helper i's effective download bandwidth being reduced to $S\mu/N + (1 - f_i)\mu$. On the one hand, helper i cannot transfer data to its collector faster than it is downloading, which leads to the constraint $f_i\mu \leq S\mu/N + (1 - f_i)\mu$. On the other hand, the transfer speed between helper i and the collector should be maximal, which occurs when $f_i\mu = S\mu/N + (1 - f_i)\mu$, or, in other words, $f_i = (S/N + 1)/2$. We can now substitute this value of f_i in Eq. (1), which leads to:

$$\begin{aligned} d \;=\; & \frac{S}{N} \cdot \mu + \mu + \frac{1}{2} \sum_{i=1}^{h} \left(\frac{S}{N} + 1 \right) \mu = \\ & \left(\frac{S}{N} + 1 \right) \left(1 + \frac{h}{2} \right) \mu. \end{aligned} \tag{2}$$

Adding new helpers makes sense only until a collector's effective download bandwidth reaches the download link capacity, which occurs when d equals c. Replacing d with c in Eq. (2) gives us the following formula for the minimum number of helpers required to fill entirely a collector's download link, denoted by h_{opt}:

$$h_{opt} = 2 \left(\frac{cN}{(S + N)\mu} - 1 \right). \tag{3}$$

4.3. The download speedup

As the performance metric which allows us to quantify the performance gain of collaborative downloads, we use the value of the collector's *speedup* u, which we define as the ratio between the download time of a peer acting on its own and the download time of a collector supported by its helpers. As the download time is directly related to the inverse of the download bandwidth, we can equivalently define the speedup as the ratio between the bandwidth of a collector and a peer acting on its own. Using the notation introduced in Section 4.2, we find for the speedup

$$u = \frac{d}{\frac{S}{N}\mu + \mu} = \begin{cases} 1 + \frac{h}{2}, & h < h_{opt}, \\ \frac{cN}{(S+N)\mu}, & otherwise. \end{cases} \tag{4}$$

In Eq. (4), $S\mu/N + \mu$ represents the download bandwidth, as defined in Section 4.2, of a peer acting on its own. The final value of the formula for u is obtained by substituting for d the value computed in Eq. (2).

For $h < h_{opt}$, Eq. (4) is intuitive. First, adding a new helper increases the collector's speedup by a constant value, which is a consequence of the observation that each helper contributes the same amount of bandwidth (see Eq. (1)). Only the contribution of the last helper added before entirely filling the collector's available bandwidth may be restricted by the remaining empty capacity of the collector's download link. Second, adding a helper increases the speedup by 0.5 as a helper has to devote equal amounts of upload bandwidth to the collector and to the peers it is bartering with outside the collaboration.

4.4. Finding bartering partners

The formula for the speedup derived in Eq. (4) is based on a simplified model of a P2P network. In particular, this model ignores the overhead introduced by locating the chunks to download. We now extend our analytical model to show that the more helpers are involved in the download, the higher is the performance penalty caused by the difficulties in locating appropriate chunks to download. Consequently, we prove that in reality the maximal speedup cannot be achieved with only h_{opt} helpers.

We start with introducing the intuition behind this statement. The consequence of the bartering-for-bytes mechanism built into BitTorrent is that the download speed of a peer is lower in the beginning and in the end of the download. In the early stage of the download peers have only a few chunks to offer, and are therefore not attractive as bartering partners. Similarly, near the end peers have difficulties in finding other peers that have the last few chunks they miss.

In 2Fast helpers are downloading distinct chunks of a file. The more helpers we use, the faster the download will finish, and consequently the lower the amount of chunks downloaded by each helper. The lower the amount of chunks a helper can offer, the more difficulties it has in finding bartering partners. Increasing the number of helpers in a collabo-

ration has, thus, a negative influence on the performance of an individual helper. Below we present a proof of this claim.

Using the notation introduced in Section 4.1 we are going to quantify the influence of the number of chunks possessed by a peer on the probability of finding a bartering partner. First, let's observe that peer i can barter with peer j only if i has a chunk k_1 which j is interested in, and j has a chunk k_2 which i wants. In other words, the exchange of chunks k_1 and k_2 between i and j is possible only if $m_{i,k_1}(1-m_{j,k_1})m_{j,k_2}(1-m_{i,k_2})$ equals 1.

The total number of chunk exchange possibilities B_i of peer i can be expressed as $\sum_{j,k_1,k_2} m_{i,k_1}(1-m_{j,k_1})m_{j,k_2}(1-m_{i,k_2})$. Assuming for simplicity, as in [11], that the probability that peer i possesses a (random) chunk is n_i/K, we may treat B_i as a random variable. The expected value $\mathbf{E}B_i$ of B_i is the number of possible chunk exchanges in which a peer may be involved. The computation of $\mathbf{E}B_i$ is straightforward.

$$
\begin{aligned}
&\mathbf{E}B_i = \\
=&\mathbf{E}\left[\sum_{j,k_1,k_2} m_{i,k_1}(1-m_{j,k_1})m_{j,k_2}(1-m_{i,k_2})\right] \\
=&\sum_{j,k_1,k_2}\mathbf{P}[m_{i,k_1}=1]\mathbf{P}[m_{j,k_1}=0]\mathbf{P}[m_{j,k_2}=1]\mathbf{P}[m_{i,k_2}=0] \\
=&\sum_{j,k_1,k_2}\frac{n_i}{K}(1-\frac{n_j}{K})\frac{n_j}{K}(1-\frac{n_i}{K}) \\
=&\frac{n_i(K-n_i)}{K^2}\sum_j\sum_{k_1,k_2}\frac{n_j(K-n_j)}{K^2} \\
=&\frac{n_i(K-n_i)}{K^2}\sum_j(Kn_j-n_j^2) \\
=&\frac{n_i(K-n_i)}{K^2}(K^2L-\sum_j n_j^2) \\
=&n_i(K-n_i)(L-\sum_j(\frac{n_j}{K})^2). \qquad (5)
\end{aligned}
$$

In the above computations we assume that the value of m_{i,k_1} is independent of the value of m_{j,k_2} for $i \neq j$ and $k_1 \neq k_2$.

Note that the value of $\mathbf{E}B_i$ is maximal when n_i equals $K/2$. This observation accords with our intuition. Namely, a peer has the biggest chance of finding a bartering partner when the number of chunks the peer still misses equals the number of chunks it can offer in return. Furthermore, $\mathbf{E}B_i$ is influenced by the distribution of chunks on other peers in the system in the form of the sum $\sum_j(n_j/K)^2$. The value of $\mathbf{E}B_i$ is maximized when the chunks are distributed evenly, i.e., every peer has the same number of chunks.

Assuming that each helper in the collaboration obtains during the entire download the same number of chunks, helpers finish the download with fewer than half $(K/2)$ of the chunks. The number of bartering possibilities of a helper behaves on the interval $[0, K/2]$ roughly as a quadratic, increasing function of the number of downloaded chunks. The more helpers are involved in the collaboration, the fewer chunks each of them downloads, the lower the number of bartering possibilities of each individual helper. The number of bartering possibilities of helpers affects their bartering efficiency, which has a negative influence on the achieved speedup of the collector. In other words, the larger the collaboration, the lower the contribution of each member of this collaboration.

5. Protocol extensions

Taking into account the conclusions of the analysis preformed in Section 4, we further extend the basic 2Fast protocol with the following two mechanisms.

Redundant chunks download. Helpers constantly monitor their download bandwidth and use a simple exponential smoothing algorithm to predict their download bandwidth usage in the immediate future. If the predicted values are lower than a certain, globally defined, constant fraction of their download link capacity, helpers start requesting chunks that have already been downloaded by the collector or by other helpers. These redundant chunks make the helpers more attractive as bartering partners by increasing their chunk exchange possibilities as explained in Section 4.4. To prevent a protocol abuse, helpers are not allowed to request chunks from each other. As the helpers are not explicitly rewarded for the download of redundant chunks, the amount of bandwidth invested in fetching redundant chunks depends of the level of altruism of a particular helper.

Sharing of swarm information. A well known property of BitTorrent-like systems is the slow-start phase. Peers in BitTorrent-like system discover the network gradually, periodically contacting a tracker [3] service. The tracker is, however, a central component which often becomes overloaded. Consequently, it may take tens of minutes before peers can bootstrap to the network and find appropriate bartering partners. To tackle this problem, we enable a collector and its helpers to share between each other the information about the IP addresses and the chunks possessed by other peers in the swarm. The information on a new peer discovered by the collector or one of the helpers is immediately propagated to all other peers in the collaboration.

6. The evaluation of 2Fast

In this section we present the results of the evaluation of the 2Fast system in both simulated and real-world environments.

6.1. The experimental setup

We have integrated the 2Fast protocol with one of the popular BitTorrent clients creating a fully functional application [10] which can profit from the collaborative downloads feature. Our implementation is backward compatible with the current BitTorrent protocol specification, meaning

that 2Fast peers can transparently connect to standard BitTorrent peers. The source code of the 2Fast client is freely available at `http://tribler.org`.

We have evaluated the performance of 2Fast on both real-world and artificially created swarms. The reason for creating artificial swarms is that some of the experiments require full control over the characteristics of all peers in the swarm, which is of course not possible in the case of real-world swarms. For the purpose of simulating BitTorrent swarms in an isolated, local environment we have developed a simulator which can integrate any BitTorrent-compatible client software. In our experiments, non-collaborating peers in the swarm are using BitTornado[1], one of the currently most popular BitTorrent clients.

The parameters of the simulation are selected to mimic a typical BitTorrent swarm. These properties are extracted from the traces of PirateBay[2], which is as of February 2006 the biggest BitTorrent tracker used by millions of users every day. Analyzing the PirateBay traces we compute the numbers of seeders and leechers in an average BitTorrent swarm, which are 10 and 100, respectively. Peers are distributing a file of size 700MB — the average file size reported by PirateBay, which is also the size of a CD image. Peer download and upload bandwidths represent the standard end-user ADSL package offerings around the world.

In the real-world experiments we connect our collaborating peers with real BitTorrent swarms. Although we do not have the control over the sizes of these swarms, the large amount of files with different popularities registered at PirateBay gives us a wide range of swarm selection options. By omitting swarms in a flash crowd [9] we can also guarantee a certain level of swarm size stability for the duration of the experiments.

6.2. The download speedup

In the first series of experiments we investigate the influence of the number of helpers on the download performance. In particular, we are interested in finding out how well our protocol approximates the maximum achievable speedup predicted by the formulas derived in Section 4.

In both the simulated and the real environments, the size of the distributed file is around 700MB. The upload and download link capacities of the collector and helpers are 256 kbps and 1024 kbps, respectively. The link bandwidth characteristics of the non-collaborating peers in the simulated swarm are the same as those of the collector and the helpers. In the real environment we do not have any influence on these values. During the simulation we keep the number of seeders and leechers constant and equal to 10 and 100, respectively. The real swarm is selected to have a similar numbers of peers of both types as the simulated swarm.

[1] `http://www.bittornado.com`
[2] `http://thepiratebay.org`

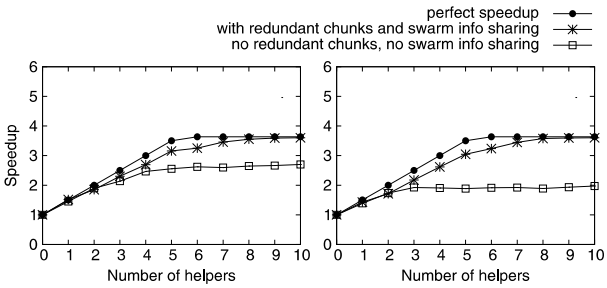

Figure 2. The speedup of 2Fast in a simulated (left) and a real (right) environment.

Figure 2 shows the speedup of our 2Fast protocol in simulated and real environments. The three lines in these plots represent the theoretical bound for the speedup, and the performance of the 2Fast protocol with and without the redundant-chunk download and swarm-information sharing optimizations described in Section 5.

The results show that the speedup achieved by the 2Fast protocol is less than 10% lower than the computed perfect speedup. The shapes of the speedup curves are similar for both the simulated and real environments, which indicates that the simulation methodology approximates well the real BitTorrent environment. The improvement gained by enabling the redundant-chunk download and swarm-information sharing features is more significant in the real than in the simulated environment. We suspect that this difference is caused by the following two facts. First, in a real environment, where bandwidth is expensive, peers are more concerned about fairness of their resource usage and set hard policies enforcing a certain sharing ratio (ratio between number of uploaded and downloaded chunks). Peers in our simulation do not restrict their upload speed as long as they have available bandwidth, and therefore the effect of the redundant-chunk download is less visible in comparison with the real environment. Second, trackers in the real environment serve not one, as in our simulation, but hundreds of swarms simultaneously and often become overloaded. The swarm-information sharing optimization was designed for situations in which the overloaded tracker slows down the bootstrapping and peer discovery process.

6.3. The link characteristics

In the second experiment we quantify the influence of a peer's link bandwidth characteristics on the performance of the collaborative downloads. The results of such an experiment can be reliable only under the assumption that all peers in the swarm have links with certain properties. This requirement excludes the possibility of using real world BitTorrent swarms. Instead, with the method described in Section 6.1, we have built a local BitTorrent swarm where all peers have fixed upload and download link capacity limits. The size of the distributed file and the number of seeders and helpers are the same as in the experiment described in

Upload/download bandwidth [kbps]	Speedup		Optimal number of helpers	
	Theoretical	Measured	Theoretical	Measured
682/1024	1.36	1.27	1	1
512/1024	1.82	1.72	2	2
256/1024	3.64	3.25	6	7
128/1024	7.27	6.4	14	17

Table 1. Efficiency of 2Fast for different upload/download link bandwidth setups.

Section 6.2.

The results of the experiment are summarized in Table 1. For each upload/download link capacity limit we compute the minimum number of helpers that should suffice to fill entirely the download link of the collector. Having this number of helpers ideally should result in theoretical perfect speedup. In reality, however, different factors such as slow-start phase and difficulties of helpers in finding bartering partners (see Section 4) result in the actual measured speedup being lower than the perfect speedup. Still, by adding a few more helpers a speedup close to the perfect one can be achieved. The last column of Table 1 gives the minimal number of helpers needed to reach a speedup which is at most 5% lower than the perfect speedup.

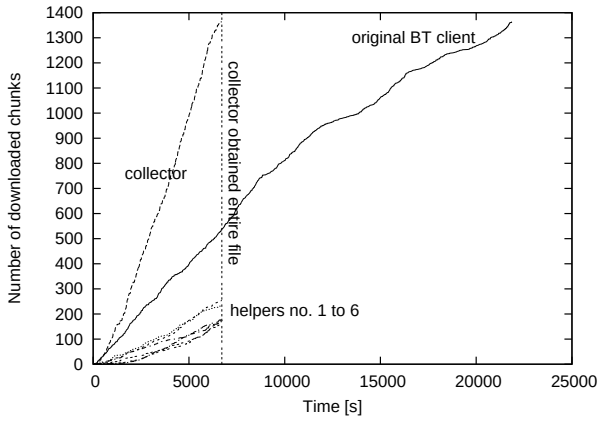

Figure 3. The download progress of a collector, its helpers and a BitTorrent peer.

6.4. The download progress

The purpose of the next experiment is to measure the download bandwidth variation over time from peers of different roles. To conduct this experiment we select an average size BitTorrent swarm hosted at PirateBay and connect to it at the same time a collector with six helpers and an original BitTorrent client downloading on his own. All our peers have the same bottleneck link bandwidth characteristics which are 256 kbps up and 1024 kbps down, and the size of the downloaded file is around 700MB. The reason of adding an ordinary BitTorrent peer instead of selecting one of the real-world peers is that this way we can enforce certain bandwidth settings and easily track the progress of all our peers. This way we can compare the download behavior of a peer with and without helpers.

Figure 3 presents the numbers of chunks downloaded by peers of different roles. Note that the helpers disconnect as soon as the collector has obtained the entire file.

Figure 4 shows the number of chunks that are uploaded to the collector by the different peers (helpers as well as ordinary BitTorrent peers) at every stage of the download. The stages are defined as 100 second intervals. The plot presented here is a stacked plot, which means that the number of the chunks contributed by a particular peer is proportional to the area of the corresponding layer. There are in total seven layers, the bottom six of which quantify the contribution of the helpers. The top layer represents the number of chunks downloaded by the collector from non-helpers. Note that the experimental results confirm the conclusions from the theoretical analysis performed in Section 4.3, which says that the download speed of a peer is fastest around the middle of the download process.

6.5. The swarm characteristics

In the last series of experiments, we investigate the influence of the swarm size characteristics on the 2Fast protocol performance. During these experiments, the collector and its helpers are connected to real BitTorrent swarms. Because we are working with real BitTorrent swarms, we do not have influence on the bandwidth characteristics of the peers in these swarms. While selecting particular swarms for our experiments, we have limited ourselves to the swarms of peers distributing files of size around 700MB, with an accuracy of 100MB. To compute the speedup of the 2Fast protocol we use the same method as in the previous experiments — in parallel with a collector and its six helpers, we start an ordinary BitTorrent client and compare the download times measured for the collector and the original BitTorrent peer. The upload and download link capacities of our peers are the same as previously.

Figure 5 (left) shows the relation between the swarm size and the value of the 2Fast protocol speedup. The swarm sizes vary from 17 to 3014 (note the logarithmic scale on the horizontal axis). It can be clearly seen that the collaborative download protocol performs much better for middle- to large-size swarms than for small swarms. In small swarms, where each chunk is possessed by only a few peers, the specific interests of the helpers make it difficult for them to find bartering partners.

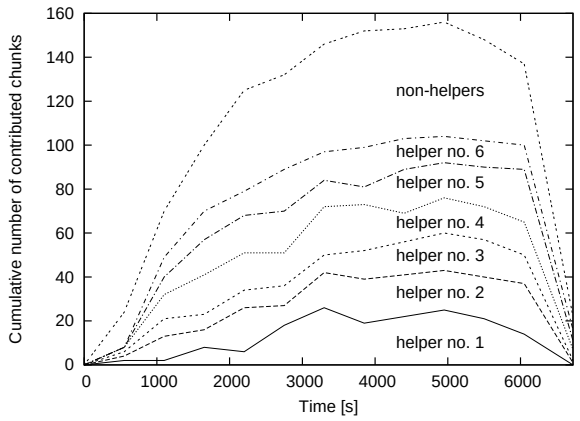

Figure 4. The numbers of chunks (measured in numbers of chunks per 100 sec intervals) downloaded by the collector from its helpers and other BitTorrent peers.

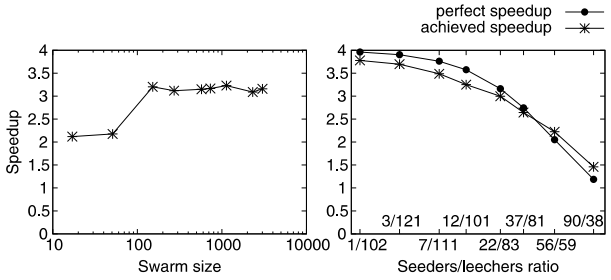

Figure 5. The influence of the swarm size on the speedup (left), and the influence of the seeders-to-leechers ratio on the speedup (right).

In the following experiment we examine the performance of the 2Fast protocol on swarms of sizes around 100 peers but with different numbers of seeders and leechers. The results are presented in Figure 5 (right). Also in this case the experimental results are consistent with the formal analysis. The value of the speedup described by Eq. (4) decreases when the seeders-to-leechers ratio increases; when there are many seeders, the potential benefit of helpers is limited. An interesting observation can be made for swarms with a high relative number of seeders. For those swarms the performance of the 2Fast protocol exceeds the theoretically computed maximum. We explain this behavior by the differences in the seeding protocols in different BitTorrent clients. Many popular BitTorrent clients implement a so-called *super seeding* algorithm which strictly limits uploading of duplicate chunks[3]. The super seeding mechanism limits the bandwidth contributions of the seeders. Consequently, in swarms with enough leechers, the number of seeders has less influence on the overall download performance then when super seeding is disabled.

[3] `http://wikipedia.org/wiki/Super_seeding`

7. Conclusions and Future Work

In this paper we have presented a novel concept of data distribution in P2P networks based on establishing collaborations of peers helping each other in downloading a single copy of a file. Our 2Fast system eliminates the bottleneck of the upload link capacity limiting the download speed of peers with asymmetric links present in other P2P download protocols. We find that, by removing this limitation, 2Fast improves the download speed by up to a factor of 3.5 in comparison to state-of-the-art P2P download protocols.

In the future we plan to extend the bandwidth sharing model introduced by 2Fast system by allowing collaborations to have more than one collector. We will also investigate different approaches for maintaining persistent credits between peers helping each other.

References

[1] E. Adar and B. A. Huberman. Free riding on gnutella. Technical report, Xerox PARC, August 2000.

[2] K. G. Anagnostakis and M. B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *ICDCS'04*, Tokyo, Japan, March 2004.

[3] B. Cohen. Incentives build robustness in bittorrent. In *P2PEcon'03*, Berkeley, CA, May 2003.

[4] G. Davies. *A history of money from ancient times to the present day*. Cardiff University of Wales Press, 2nd edition, 1996.

[5] G. Hardin. The tragedy of the commons. *The Science*, 162:1243–1248, December 1968.

[6] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, San Jose, CA, June 2003.

[7] A. Nandi, T.-W. Ngan, A. Singh, P. Druschel, and D. S. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In *Middleware 2005*, Grenoble, France, November 2005.

[8] A. Parker. The true picture of peer-to-peer filesharing, 2004. `http://www.cachelogic.com/`.

[9] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. A measurement study of the bittorrent peer-to-peer file-sharing system. In *IPTPS'05*, Ithaca, NY, February 2005.

[10] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A social-based peer-to-peer system. In *IPTPS'06*, Santa Barbara, CA, February 2006.

[11] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM*, Portland, OR, August 2004.

[12] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network, 2001.

[13] S. Saroiu, K. P. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *MMCN'02*, San Jose, CA, January 2002.

[14] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. In *IPTPS'03*, February 2003.

[15] M. Yang, Z. Zhang, X. Li, and Y. Dai. An empirical study of free-riding behavior in the maze p2p file-sharing system. In *IPTPS'05*, Ithaca, NY, February 2005.