Collaboration in BitTorrent Systems

Rafit Izhak-Ratzin

Computer Science Department, University of California - Los Angeles
(310) 825-3886, 4732 Boelter Hall, Los Angeles, CA 90095
rafiti@cs.ucla.edu

Abstract. Recent research efforts have shown that the popular BitTorrent protocol does not strictly enforce fairness and allows free-riding, mainly via optimistic unchokes.

This paper proposes a BitTorrent-like protocol, that encourages peers of similar upload bandwidth to be *buddies*— peers collaborating for mutual benefit. Buddy peers mostly satisfy their download needs through their buddies and perform optimistic unchokes only when absolutely necessary. As a result, the buddy protocol improves fairness via explicit cooperation between buddies, and limits bandwidth spent on random optimistic unchokes, leading to a system more robust against free-riders.

We implemented the buddy protocol on top of an existing BitTorrent implementation and ran experiments on a controlled PlanetLab testbed to evaluate its impact. Our results show that the buddy protocol promotes fairness, discourages free-riding, and improves the robustness of the system as compared to regular BitTorrent. It also provides incentives to be adopted by all the peers in the system.

Keywords: BitTorrent, Buddies, fairness, incentives.

1 Introduction

The BitTorrent protocol [1], one of the most popular protocol for peer-to-peer content distribution, aims to provide fairness among peers by applying the tit-for-tat (TFT) unchoking mechanism to incentivize contribution and discourage free-riding. Despite this, previous studies [2, 3, 4] showed that BitTorrent suffers from unfairness, particularly for high capacity leechers. Other studies [5, 6, 7] showed that free-riding opportunity exists in BitTorrent. One of the main contributor to this lack of fairness is the *optimistic unchokes* mechanism, another unchoking mechanism that is used in BitTorrent. As this mechanism facilitates continuous discovery of better peers to interact with, it also facilitates upload imbalance, dynamically creates a major opportunity for peers to obtain data without uploading in exchange. However, it is also been shown to greatly contribute to the robustness of the protocol [5], making it hard to be replaced.

In this paper we propose an alternative mechanism that dynamically creates *buddies*—pairs of peers having similar upload bandwidth that collaborate for mutual benefit. The buddy relation is based on the upload history during the

© IFIP International Federation for Information Processing 2009

relation existence, and not only on the last round of contribution as in regular BitTorrent's TFT mechanism. Peers in buddy mode aim to satisfy their download needs through their buddies and perform optimistic unchokes only if absolutely necessary. Hence, the buddy mechanism mostly obviates the need for optimistic unchokes, while at the same time improves fairness and system robustness.

We implemented our buddy protocol on top of an existing BitTorrent implementation. We ran experiments with different configurations on controlled PlanetLab testbed in order to evaluate the impact of the buddy protocol on different level of contributing leechers and free-riders in comparison to the regular BitTorrent. In the experiments, our buddy protocol exhibits the following properties:

1. It promotes fairness as high capacity leechers who suffer from the unfairness in regular BitTorrent improve their download rate, and low capacity leechers who used to benefit from this unfairness are slowed down.
2. It provides a clear incentive for all level of contributing leechers to adopt it, even to those low capacity leechers who are slowed down due to it.
3. It discourages free-riding by limiting the optimistic unchokes, in comparison to the regular BitTorrent, thus directly hurting free-riders performance.
4. It improves the system robustness as increasing free-riding has less impact on the contributors performance, as compared to the regular BitTorrent.

We develop an analytical model that explains leechers interactions in the presence of the buddy protocol and justifies our design choices. Based on this model and game theory, we prove the existence of Nash Equilibrium when all the contributing leechers adopt the buddy protocol.

The rest of this paper is structured as follows. In section 2 we provide a brief description of BitTorrent. In section 3 we present the design of our buddy protocol, while in section 4 we describe the analytical model that has guided this design. Implementation details are discussed in Section 5, and then the results of our PlanetLab experiments are presented in Section 6. Finally, section 7 sets our approach in the context of related work, and section 8 concludes.

2 BitTorrent Overview

The BitTorrent protocol is a popular peer-to-peer content distribution protocol that has been demonstrated to scale well with many participating clients.

Before the distribution process, the content provider divides the data content into multiple *pieces* which further divided into multiple *subpieces*. It then creates a *metainfo file*, containing information necessary for initiating the download process. This information includes the address of the *tracker*, a centralized coordinator that facilitates peer discovery.

In order to join a *torrent*(or swarm)—a group of peers participating in download of a content—a client downloads the metainfo file. It then connects the tracker from which it receives a *peer set* of randomly selected peers currently transferring the content pieces. These might include both *leechers*, who are still in the downloading process, and *seeds*, who have the entire content and are providing it to others. The new client can then connect peers in this set and request pieces.

The decision to whom to upload data is made independently by the leecher via the *choking algorithm*, applying the *tit-for-tat* mechanism that gives preference to those leechers who upload data to the given leecher at the highest rate. More specifically, once every *rechoke period*, typically 10 seconds, a leecher checks the current download rates of all leechers in its peer set. Then it selects the fastest uploaders and only uploads to those, while choking the rest for the duration of the rechoke period. This unchoking mechanism is called *regular unchoke*. The number of unchoked peers (slots) depends on the implementation and might be fixed or changed dynamically as a function of the available upload bandwidth. A different unchoke strategy is followed by seeds since they do not need to download any pieces. The most common one is unchoke leechers in a round-robin aiming to distribute data uniformly. We used this strategy in our experiments.

Beside the regular unchokes, leechers reserve a portion of their available bandwidth for sending pieces to random peers, a so-called *optimistic unchoke* mechanism. This mechanism enables the continuous discovery of better partners, and it also enables newcomer leechers, to download some data and start exchanging pieces. Optimistic unchokes are rotated randomly among the peer set typically once every three rechoke periods allowing enough time for leechers to demonstrate cooperative behavior.

2.1 Motivation

In this section we look at some problems in BitTorrent that motivate our design. Figure 1 shows the impact of optimistic unchokes on the expected download rate as a function of the P2P observed bandwidth distribution given in [2]. We assume that one unchoke slot is used by the optimistic unchokes mechanism, and that the regular unchoke mechanism works perfectly, i.e. the download rate and the upload rate through regular unchoke are equal. Clearly, we can see that the sub linear behavior of the expected download rate leads to unfairness for high capacity leechers, where low capacity leechers benefit from this unfairness. Note that in the observed bandwidth distribution the majority of the leechers are low capacity leechers with 88% of the leechers having less than 300KB/s upload capacity. This result motivates us to look for a replacement for the optimistic unchoke mechanism.

As for the regular unchoke mechanism, the number of unchoke slots that a leecher uses for regular unchoke can be fixed or dynamic as a function of the leecher's ability to saturate its upload capacity. This again might lead to unfairness, since typically, in real torrents, the download capacity of a leecher might be greater than the upload capacity. Thus, high capacity leechers might upload in full capacity, but not be able to download as much due to upload constraints of the downloading leechers and limited number of unchoke slots.

In addition, the choking decision in based on leechers' upload rate from the last rechoke period only, ignoring longer history that is available.

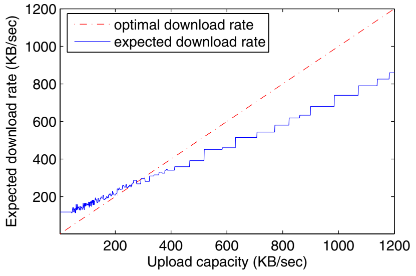


Fig. 1. Expected download rate vs. upload capacity for partial bandwidth spectrum

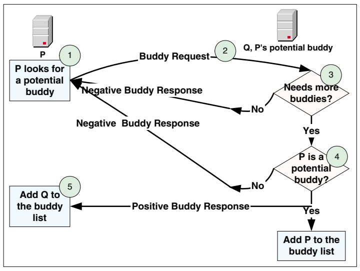


Fig. 2. Buddy formation mechanism

3 Design

We now turn to describe the design of our buddy protocol, which augments BitTorrent with the notion of *buddies*—pairs of peers having similar upload capacity, collaborating for mutual benefit. The main goals of the protocol are reducing the unfairness that BitTorrent suffers from, and discouraging free-riding. The protocol achieves these goals via explicit cooperation, as well as the tit-for-tat choking mechanism already employed by BitTorrent. In addition, it significantly limits the optimistic unchokes, where high capacity clients are forced to peer with those with low capacity while searching for better peers, an acknowledged opportunity for free-riding in BitTorrent [7].

The protocol comprises three main parts: the process by which buddy relations are formed, the process by which buddy relations are monitored, and the algorithm peers use to decide who to unchoke. We describe these in turn.

3.1 Buddy Formation

A leecher P that is running in buddy mode, is willing to maximize the number of buddies, who make it their priority to serve P, as P serves them in return. P reserves an unchoked slot to each buddy it can upload data to in order to minimize buddy chokes, which can lead to termination of buddy relation. At the same time, P reserves one unchoke slot for regular unchoke in order to avoid isolating group of buddies from the rest of the swarm. Therefore, P can have maximum $n_s - 1$ buddies, given that n_s is the number of unchoke slots in P.

Figure 2 shows the buddy formation process. P will run this process once every rechoke period, only if it has not reached the maximum number of buddies.

Step 1. Aiming to find a potential buddy with similar upload as its own, P will estimates the upload bandwidth of peers it interacted with, that are not its buddies, using its own history about past download interactions with these peers. If P is not able to find a potential buddy Q due to its upload capacity that is too high(low) compared to the leechers it has interacted with, it may

gradually increase(decrease) the number of unchoke slots n_s, while keeping maximum(minimum) n_s size constraints satisfied.
Step 2. Once P has found a potential buddy Q, P sends a buddy request to Q, asking Q to be its buddy.
Step 3. Upon receiving a buddy request from P, Q checks that it has not reached the maximum number of buddies, otherwise it sends a negative buddy response.
Step 4. If Q can have more buddies, it estimates P upload capacity. If it finds that P has similar upload capacity as its own, Q sends a positive buddy response to P and tags P as buddy. Otherwise it sends a negative buddy response to P.
Step 5. Upon receiving a positive buddy response from Q, P tags Q as buddy.

In order to keep the design simple the look-up for buddies does not require a view of all the network peers, thus, we expect the improvement to be suboptimal.

3.2 Monitoring Buddies

Every rechoke period, a peer in buddy mode checks the upload rate of its buddies. It looks separately at the history of each buddy, since the buddy connection was established, and checks that the estimated upload rate of the buddy remains similar to its own. If this is not the case, the peer removes the buddy that uploads slower than expected from its buddy list. Thus the longer the connection the more reliable it is in comparison to the regular unchoke mechanism. A buddy can not gain much from cheating by uploading less than agreed, since it will be disconnected quickly. Thus, it will not be able to download much before being caught, and will lose an established buddy connection. Notice that a stronger punishment can be considered by keeping the cheating buddies' history and banning them from download or have a buddy relation for a time.

3.3 Unchoking Decisions

During the uploading process, a leecher decides periodically which peers to unchoke. In regular BitTorrent protocol, this unchoking decision is made using two separate mechanisms, the regular unchoke mechanisms and the optimistic unchoke mechanism. The former guides a leecher to unchoke those peers that upload data to the given peer at the highest rates, while the latter selects the unchoked peers at random, independently of their contributions.

Our buddy protocol extends the two given mechanisms with a third, *buddy unchoke* mechanism, which dictates that a leecher always unchokes all its buddies. The rationale is that a leecher that has buddies should opt for uploading to its buddies whenever possible, expecting its buddies to do the same. Hence, since buddies have similar upload rate, this guarantees similar upload as download rate. At the same time, it performs at least one regular unchoke to a peer that is not its buddy to avoid isolating group of buddies from the rest of the swarm. Uploading via optimistic unchokes is performed only as a last resort, and it may happen only when a leecher has no data other buddies are missing or does not reach the maximum number of buddies. The combined unchoke decision algorithm is as follows.

Step 1 (buddy unchokes). A leecher in buddy mode strives to satisfy as many of its buddies as possible. We denote the number of unchoke slots reserved for buddies as a function of time with $n_B(t)$, where $0 \leq n_B(t) \leq n_s - 1$, $n_B(t)$ is bounded by $n_s - 1$ devoting at least one slot for regular unchokes.
Step 2 (optimistic unchokes). Given that P_0 is the probability to perform optimistic unchokes in regular BitTorrent, a leecher in buddy mode performs optimistic unchokes with probability:

$$P_B(t) = P_0 \frac{n_s - n_B(t) - 1}{n_s - 1} \tag{1}$$

Step 3 (Regular unchokes). $n_O(t)$ denotes the number of optimistic unchokes slots. Thus, for the remaining $(n_s - n_B(t) - n_O(t))$ unchoke slots, a leecher selects those peers that were not selected already, having the highest uploading rates.

Note that for independent leechers that run the regular BitTorrent $P_B = P_0$, since $n_B = 0$. However, P_B decreases for buddies as more buddies are served. Those peers who serve $n_s - 1$ buddies perform no optimistic unchokes ($n_B = n_s - 1 \rightarrow P_T = 0$). They do not need to continue peers discovery.

Based on this unchoking algorithm design, we expect the following effects:

1. High capacity buddy peers' performance should be improved, as there are now other high capacity peers who make it their priority to serve them.
2. Reduction in the total number of optimistic unchokes, thus directly hurting the free-riders, who critically depend on these unchokes to obtain data.

These hypotheses validated by our experimental evaluation results. In a sense, *BitTorrent's optimistic unchoke mechanism is mostly replaced by buddies* as the means of discovering better partners. Hence, buddies no longer need to explore the network as much, having the guarantee of other leechers with similar upload capacity willing to upload to them.

4 Analytical Model

This section presents an analytical model that guides the protocol design presented in the previous section. It describes peer interactions in the presence of buddies as a game with rational players, and shows that there is a Nash-Equilibrium when all the peers in the system are contributing buddies.

4.1 The Game

We model the process of distributing a given content between incentive leechers in the P2P system as an infinitely repeated *game* [8], with rational participants. To simplify this process, we do not consider long-term punishment, although this is a design issue that is part of future work. The players repeatedly engage in rounds. In each round they choose their actions simultaneously, where the actions are determined by the players' strategy choices. A player can choose one of the three following strategies.

1. **an independent peer:** a contributor who runs the regular BitTorrent
2. **a free-rider:** a player who does not upload data to others
3. **a buddy-aware peer:** a player who follows the buddy-enhanced protocol

4.2 Leecher Utility Function

The leecher utility function is defined as the average download rate from all other leechers, thus leecher i's utility function is $U_i = \overline{d}_i$. We will ignore the download from the seeds in the analysis, since a seed upload process does not depend on the existence of buddies.

In [9], Fan *et al.* have already expressed the leecher utility function for the regular BitTorrent protocol as a combination of the average download rate that is obtained through the regular unchoke and the optimistic unchoke mechanisms. The buddy-enhanced BitTorrent protocol adds yet another term for the average download rate that is obtained through the buddy unchoke policy. Thus, the utility function of a leecher in the buddy-enhanced BitTorrent will be

$$\overline{U}_i = \overline{d}_i(regular) + \overline{d}_i(buddies) + \overline{d}_i(optimistic) \tag{2}$$

We assume that all peer i's buddies have a similar upload rate of $\frac{u_i}{n_s}$ to i, given that u_i is i's upload rate. That is due to the way the buddy connections are created and maintained. In addition, we assume that the download bandwidth of a leecher is at least as much as its own upload bandwidth; this is typically the case in real torrents.

Each peer that has n_B buddies enjoys a download rate of $\frac{u_i}{n_s}$ from each of its buddies. As a result, the buddy policy yield will be

$$\overline{d}_i(buddy) \approx n_B \cdot \frac{u_i}{n_s} \tag{3}$$

$P_{db} = \frac{n_B}{n_s}$ describes the download rate a peer enjoys from its buddies, as it also represents the percentage of upload bandwidth it spends on those buddies.

In [9], Fan *et al.* have shown mathematically that the download bandwidth of a leecher is approximately equal to its upload bandwidth, for regular unchokes, i.e., $d_i \approx u_i$. The percentage of the upload bandwidth dedicated to regular unchokes is $1 - P_{db} - P_B$. Thus, the regular unchoke yield will be

$$\overline{d}_i(regular) \approx (1 - P_{db} - P_B) \cdot u_i \tag{4}$$

Therefore, from Equations (2), (3), and (4), the utility function of buddy i is:

$$U_i \approx (1 - P_B) \cdot u_i + \overline{d}_i(optimistic). \tag{5}$$

The utility function of a free-rider who uploads nothing is:

$$U_{FR} = \overline{d}_{FR}(optimistic) \tag{6}$$

The utility function of an independent peer who runs the regular BitTorrent is:

$$U_{IL} \approx (1 - P_0) \cdot u_i + \overline{d}_{IL}(optimistic). \tag{7}$$

Note that since the download through optimistic unchokes can be received from any peer in i's peer set as it is based on random selection among all peers, the optimistic unchokes yield is not sensitive to the strategy choice of peer i. Hence, the unfairness in the regular BitTorrent is a function of the upload capacity and P_0, and it appears when the amount of upload that is given to optimistic unchokes is not the same as the amount of download that is received through optimistic unchokes, i.e., $P_0 \cdot u_i \neq d_i(optimistic)$.

In the buddy-enhanced BitTorrent, the unfairness is a function of the upload capacity, and P_B. P_B gets closer to 0, as the number of buddy connections increases, which leads to reduction in optimistic unchokes upload. Thus, peers will benefit more by maximizing the number of buddy connections. Morover, increasing in number of buddy connections in the system will reduce the average download that is received through optimistic unchokes. As for the system, if every peer will choose the buddy-aware strategy, thus minimizing P_B and devoting less upload to the unfairness inequality, the unfairness in the system will decrease.

4.3 Nash Equilibrium

Theorem 1. *The game in a buddy-enhanced BitTorrent system has a Nash Equilibrium when all the rational leechers in the game are buddy-aware peers.*

Proof. In order to prove this, we need to show that the utility of a buddy-aware peer is not lower than any other strategy's utility. Thus, if all the peers choose the buddy-aware strategy, none of the peers have the incentive to change their strategy. Equation 1 shows that $P_B \leq P_O$. Therefore, $U_i(IL) \leq U_i(\text{buddy-aware})$. Both utilities are equal when the buddy-aware peer is not able to find any matching buddy through the whole downloading process. Otherwise, the utility of the buddy-aware peer will be always greater than the utility of an independent peer. A free-rider's utility will be $U_i(FR) \leq U_i(\text{buddy-aware})$. The utility of a free-rider will be equal to the utility of a buddy-aware i only in the case when all the peers in the network do not upload to i through the regular unchokes or the buddy unchokes policies, during the whole downloading process.

We can thus conclude that the buddy-enhanced system has a Nash Equilibrium when all the rational peers are buddy-aware peers.

5 Implementation

The analytical model shows that the buddy-enhanced protocol has the potential to reduce unfairness and free-riding in a BitTorrent system. In order to examine this claim, we modified an existing open-source BitTorrent client to implement our buddy protocol. This section discusses interesting aspects of our prototype.

Our buddy-enhanced BitTorrent client was implemented on top of Enhanced CTorrent, version 3.2 [10]. The modified client can run in *buddy mode*, or in *regular mode*, in which it simply runs the regular BitTorrent. We also added functionality for a buddy peer to maintain state about its buddies.

Rechoke period in Enhanced CTorrent takes 10 seconds, and optimistic unchokes are rotated every 3 rechoke periods. The number of unchoke slots is dynamic with a minimum of 4 slots. This number may increase if a client does not saturate its capacity. In buddy mode, this number might change also if the upload capacity of a peer is too low/high compared to the peers it interacts with. We did not limit the maximum number of slots.

In regular mode, one unchoke slot is always devoted to optimistic unchokes, while in buddy mode, our unchoking mechanism determines the number of optimistic unchokes. In particular, every 3 rechoke periods a single optimistic unchoke is performed with probability given by Equation 1.

Before running in buddy-mode, a peer needs to collect information about the upload rate of itself and other peers in its peer set. The first 3 minutes (6 optimistic unchoke periods) after a client starts uploading are used for that, only after this period the peer is able to start running in buddy mode.

No changes were required for the tracker.

6 Experimental Evaluation

We ran extensive experiments on a controlled testbed, validating our protocol properties. Here we discuss our experimental methodology and results.

6.1 Methodology

All our experiments were ran on the PlanetLab experimental platform [11], utilizing nodes that are located across the globe. We executed all experiment runs consecutively in time on the same set of machines. Unless otherwise specified, we use the default BitTorrent client and seeding behavior.

All peers started the downloading process simultaneously, emulating a flash crowd scenario. The initial seeds stayed connected through the whole experiment. Leechers left the network once they have downloaded the entire content.

We artificially constrained the upload capacity of the nodes according to the bandwidth distribution for typical BitTorrent leechers as presented in [2]. This distribution is based on empirical measurements of BitTorrent swarms including more than 300,000 unique BitTorrent IPs. Not all nodes were capable of matching the required upload capacity that drawn from the empirical distribution. Thus, we scaled by 1/20th the upload capacity and other relevant experimental parameters such as file size. We did not define any download limits.

Unless otherwise specified, the experiments host 104 PlanetLab nodes, 100 leechers and 4 seeds with combined capacity of 128 KB/s serving a 30 MB file.

6.2 Results

We look at comparative results for leechers with different contribution levels, in order to validate our design.

Leecher Performance - System without free-riders. In this subsection we discuss two aspects that have been raised in this study: (1) Does the buddy-enhanced protocol indeed promote fairness? (2) Do all strategic leechers (having different upload capacity) have incentives to adopt the buddy-enhanced protocol.

Figure 3 shows leechers' download completion time comparing a regular BitTorrent system to a buddy-enhanced system where all leechers run in buddy mode. For each group of leechers having similar upload capacity, we draw separate boxplots [12] for the different scenarios. The top and the bottom of the box represent respectively the 75th and the 25th percentile download rates over all 7 runs of the experiment. The marker inside the box represents the median, while the vertical lines extending above and below the box represent respectively, the maximum and minimum values within the ranges of 1.5 time the box height from the box boarder. Potential outliers are marked individually with "+" sign.

We observe a clear difference between high capacity leechers which are the fastest 40% leechers and low capacity leechers which are the slowest 60% leechers. High capacity leechers improved their download time tremendously. Leechers with upload capacity of at least 5kB/sec improved their download time by 9%-35% as measured by the median. This happened due to downloading consistently from leechers with similar upload rate, as well as limiting in optimistic unchokes, and increasing number of unchoke slots for high capacity leechers.

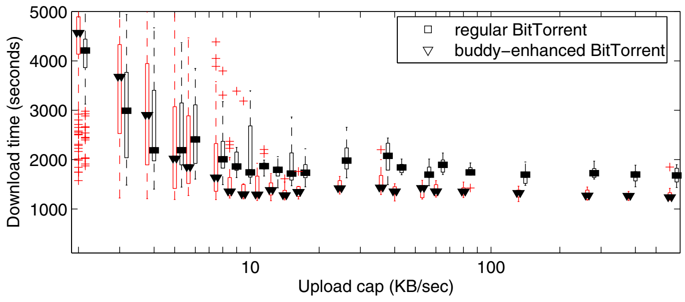
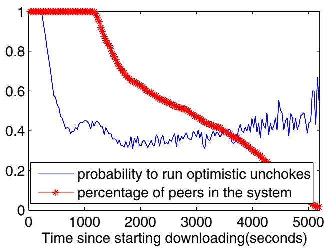


Fig. 3. Download completion time for leechers



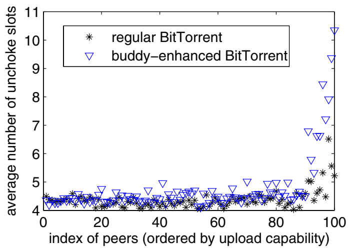


Fig. 4. The probability for optimistic unchokes in buddy-enhanced system

Fig. 5. The average unchoke slots for leechers ordered by upload capacity

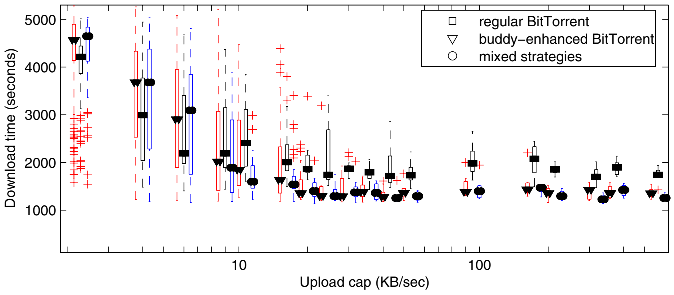


Fig. 6. Download completion time for leechers

Figure 4 confirms the reduction in optimistic unchokes by showing the average percentage of optimistic unchokes performed in the buddy enhanced system as a function of optimistic unchokes time periods. In regular BitTorrent, leechers run the optimistic unchokes mechanism every optimistic unchokes period, thus, this percentage value always equal to 1. In the buddy enhanced protocol, the optimistic unchokes reduced by $\sim$ 60% for most of the downloading process. Figure 5 confirms the increasing number of unchoke slots for the fastest 10% leechers in the buddy-enhanced protocol.

As a result of the aforementioned, low capacity leechers downloaded less from high capacity leechers, slowing down by 9%-32% as shown in Figure 3.

In summary, we see that *the buddy-enhanced protocol indeed promotes fairness and provides clear incentives for high capacity leechers to run in buddy mode*, as a high capacity buddy leecher achieves improved performance compared to an independent one. However, it is not clear that the low capacity leechers have the same incentives, since their download time performance suffers.

Hence, we ran another experiment, where low capacity peers ran in independent mode, and high capacity peers ran in buddy mode. Figure 6 that compares our previous results with the "mixed strategies" results shows that by changing strategy to independent, low capacity peers suffered even more, since as independents they were also losing part of their bandwidth for optimistic unchokes without getting anything in return. Moreover, the high capacity peers performance were not

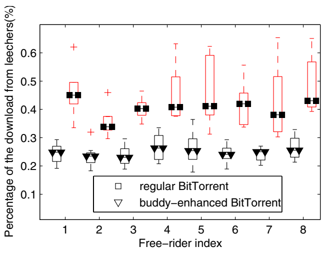


Fig. 7. FRs' download from leechers

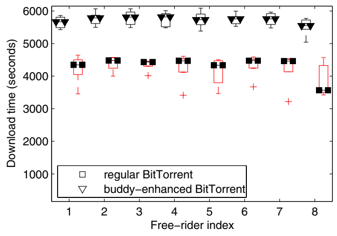


Fig. 8. FRs download time

very sensitive to the change, most of them slightly improved their performance in comparison to all are buddies results. This is because in the mixed strategies experiment high capacity peers enjoy increasing amount of optimistic unchokes from low capacity peers needing to give nothing in return.

The experiments validate our hypothesis in section 4 showing that it is to the benefit of all the leechers to run in buddy mode. Furthermore, running in buddy mode motivates leechers to contribute and improves fairness in the network.

Leecher Performance - System with Free-rider. As this section focuses on the impact of free-riders, we added 8 free-riders , having a total of 112 peers in the experiment: 100 contributors and 4 seeds as described in previous experiments and 8 free-riders that download as much as possible, but do not upload any data.

With free-riders in the network high capacity leechers in buddy mode improved their download time performance by 17%-54% as measured by the median in comparison to regular BitTorrent, while low capacity leechers were slowed down by 16%-59%. We got similar results to those shown in Figure 6, however, the different between the two protocols performance is more pronounce. That is since contributors in buddy mode limit the optimistic unchokes, hence limit the possibility to be affected by free-riders. Figure 7 validates this hypothesis, showing that the percentage of content free-riders download from leechers decreases from $\sim 45\%$ in regular BitTorrent to $\sim 25\%$ in buddy-enhanced system.

Figure 8 shows the download completion times of free-riders, showing that *the existence of buddies slowed down free-riders by 29-56%.* As in Figure 7, this is because free-riders can not take advantage of optimistic unchokes as in regular BitTorrent and thus have to depend mainly on the seeds for downloading data.

These results also indicate the increased robustness of the system to free-riding. We also investigate the system robustness by increasing the number of free-riders in the system to 16, 20, 24, however, due to lack of space we can not present in detail these results. In summary, these results show that the buddy enhanced system is less sensitive to the increasing number of free-riders, an indication to a more robust system.

7 Related Work

There is large amount of works that model the behavior and analyze the performance of BitTorrent, since Bram Cohen, the protocol's creator, first described the protocol's main mechanisms and their design rationale [1].

In [13], Qiu *et al.* presented a fluid analytical model of BitTorrent systems. They studied the choking algorithm and its effect on peer performance. They indicated that the optimistic unchokes mechanism may provide a way for leechers to free-ride. More recently, in [9], Fan *et al.* model BitTorrent capturing the fundamental trade-off between performance and fairness. Our analytical model is inspired by their work. Furthermore, in [14], Legout *et al.* observed that the incentives of tit-for-tat tend to cause leechers to cluster together with other similar upload bandwidth leechers. Our protocol facilitates this explicitly by having buddies, peers with similar bandwidth to collaborate.

Other researchers have studied the feasibility of free-riding behavior, when peers download without uploading, attempting to deceive the protocol mechanisms. The first to pinpoint that effective free-riding in BitTorrent is feasible were Shneidman *et al.* [15]. Liogkas *et al.* [5] designed and implemented 3 exploits that allow free-riders to obtain high download rates under specific circumstances. Sirivianos *et al.* [7] showed that maintaining a larger-than-normal view of the system, provides a free-rider a much higher probability to receive data from seeds and via optimistic unchokes. Finally, in [2], Piatek *et al.* also observed the unfairness. They proposed a new choking mechanism that reallocates upload bandwidth and aims to maximize peer download rates. The aforementioned studies identify one of the most important vulnerabilities that our work directly addresses, namely exploiting optimistic unchokes to download data for free.

Researchers [16, 17, 18, 19] have recognized the value of peers cooperating to download content. However all these works do not address the incentives of the volunteer peers to voluntarily download popular content pieces and upload them to others. Our proposed protocol, on the other hand, addresses this by providing a clear incentive for leechers to increase their buddy connections and upload to their buddies. This leads to reduction in enforced altruism and therefore improves fairness, and limits free-riding.

The most related protocol to our proposed protocol is the protocol that embeds teams [20], groups of peers that collaborate to improve their upload rate. There is a trade-off between the team protocol and the presented buddy protocol, which is a simplicity for the price of less improvement. The main different is that teams are formed and managed by the tracker, a centralized authority, therefore it achieves a faster convergence and a larger improvement in fairness and free-riding reduction. However, in the presented work the tracker does not involve in the buddy related part of the protocol. Hence, here we suggest a more distributed protocol, involves less management overhead, and is simpler to implement, which make it more realistic to be adopted.

8 Conclusion

In this paper we present an extension to the BitTorrent protocol, an alternative buddy mechanism that relies on continuous history to match similar upload capacity peers. It mostly replaces the optimistic unchokes, partially replaces the original TFT mechanism, and add more dynamic to the number of unchoke slots. Experimental results on a controlled testbed demonstrate that our buddy protocol promotes fairness compared to the regular BitTorrent, and provides clear incentives to be adopted, even by those low capacity peers that were slowed down due to the presence of buddies. In the presence of buddies, free-riders achieve lower download rates, compared to the regular BitTorrent. Furthermore, the system with buddies is more robust being less sensitive to increasing number of free-riders in the network.

Acknowledgments

We thank Nikitas Liogkas, Rupak Majumdar, Mihaela van der Schaar, Giovanni Pau, and Cesar Marcondes for helpful discussions and comments.

References

1. Cohen, B.: Incentives Build Robustness in BitTorrent. In: P2PEcon 2003 (2003)
2. Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., Venkataramani, A.: Do incentives build robustness in BitTorrent?. In: NSDI 2007 (2007)
3. Bharambe, A., Herley, C., Padmanabhan, V.: Analyzing and improving a bittorrent network's performance mechanisms. In: INFOCOM (2006)
4. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: Measurements, analysis, and modeling of BitTorrent-like systems. In: IMC 2005 (2005)
5. Liogkas, N., Nelson, R., Kohler, E., Zhang, L.: Exploiting BitTorrent For Fun (But Not Profit). In: IPTPS 2006 (2006)
6. Locher, T., Moor, P., Schmid, S., Wattenhofer, R.: Free Riding in BitTorrent is Cheap. In: HotNets-V (2006)
7. Sirivianos, M., Park, J.H., Chen, R., Yang, X.: Free-riding in BitTorrent Networks with the Large View Exploit. In: IPTPS 2007 (2007)
8. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
9. Fan, B., Chiu, D.M., Lui, J.C.: The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design. In: ICNP 2006 (2006)
10. Enhanced CTorrent: `http://www.rahul.net/dholmes/ctorrent:`
11. Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., Wawrzoniak, M.: Operating System Support for Planetary-Scale Network Services. In: NSDI 2004 (2004)
12. Robert McGill, J.W.T., Larsen, W.A.: Variations of box plots. The American Statistician 32, 12–16 (1978)
13. Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In: SIGCOMM 2004 (2004)
14. Legout, A., Liogkas, N., Kohler, E., Zhang, L.: Clustering and Sharing Incentives in BitTorrent Systems. In: SIGMETRICS (2007)
15. Shneidman, J., Parkes, D.C., Massouliè, L.: Faithfulness in Internet Algorithms. In: PINS 2004 (2004)
16. Wang, J., Yeo, C., Prabhakaran, V., Ramchandran, K.: On the role of helpers in peer-to-peer file download systems: design, analysis, and simulation. In: IPTPS 2007 (2007)
17. Wong, J.H.T.: Enhancing Collaborative Content Delivery with Helpers. M.sc thesis, University of British Columbia (September 2004)
18. Garbacki, P., Iosup, A., Epema, D., van Steen, M.: 2Fast: Collaborative downloads in P2P networks. In: P2P 2006 (2006)
19. BTSlave protocol page, `http://btslave.sourceforge.net`
20. Izhak-Razin, R., Liogkas, N., Majumdar, R.: Team incentives in bittorrent systems. In: TR 090002, UCLA CSD