

# On the Impact of Greedy Strategies in BitTorrent Networks: the Case of BitTyrant

Damiano Carra  
EURECOM  
Sophia Antipolis, France  
carra@eurecom.fr

Giovanni Neglia  
INRIA, Sophia Antipolis, France, and  
Università degli Studi di Palermo, Italy  
giovanni.neglia@ieee.org

Pietro Michiardi  
EURECOM  
Sophia Antipolis, France  
michiard@eurecom.fr

## Abstract

*The success of BitTorrent has fostered the development of variants to its basic components. Some of the variants adopt greedy approaches aiming at exploiting the intrinsic altruism of the original version of BitTorrent in order to maximize the benefit of participating to a torrent.*

*In this work we study BitTyrant, a recently proposed strategic client. BitTyrant tries to determine the exact amount of contribution necessary to maximize its download rate by dynamically adapting and shaping the upload rate allocated to its neighbors. We evaluate in detail the various mechanisms used by BitTyrant to identify their contribution to the performance of the client.*

*Our findings indicate that the performance gain is due to the increased number of connections established by a BitTyrant client, rather than for its subtle uplink allocation algorithm; surprisingly, BitTyrant reveals to be altruistic and particularly efficient in disseminating the content, especially during the initial phase of the distribution process. The apparent gain of a single BitTyrant client, however, disappears in the case of a widespread adoption: our results indicate a severe loss of efficiency that we analyzed in detail. In contrast, a widespread adoption of the latest version of the mainline BitTorrent client would provide increased benefit for all peers.*

## 1 Introduction

BitTorrent [1] is a peer-to-peer (p2p) content distribution application that has been adopted by millions of end-users, as witnessed by several specialized sources [2, 3, 4]. BitTorrent (BT) has not only gained a huge popularity among the mass, it has also attracted the attention of a large body of researchers that focused on its building blocks and its performance analysis through measurement [5, 6, 7], simulation [8, 9] and analytical [10, 11, 12] studies. These previous

works indicated that the key of its success can be substantially attributed to its scalability and its greater robustness to free-riding in comparison to previous p2p proposals.

Some recent studies [13, 14, 15] have proposed new clients, that are compliant to BitTorrent message protocol, but change its algorithms and adopt greedy strategies with the purpose to optimize the local performance of the client. For example, authors in [14] designed a modified client, called BitThief, that tries to maximize the client download rate without uploading any content by continuously increasing its neighborhood set. Another prominent example is that of BitTyrant [15], which tries to maximize its download rate by shaping its contribution to remote peers; additionally, BitTyrant borrows the technique to construct large neighborhoods from BitThief. Note that while BitThief client is intrinsically a free-rider, BitTyrant makes its whole upload capacity available to spread the content. Similar techniques are proposed in [13].

Our research interest is twofold. First, we want to evaluate to what extent greedy clients have competitive advantages in comparison to standard ones and hence can be expected to be widely adopted by the peer-to-peer community. Second, we want to investigate if the widespread adoption of such techniques would lead to a general performance improvement (as it is suggested in [13, 15]). We first focus on a single client (we chose BitTyrant in this work because it merges several greedy techniques discussed in the literature) and characterize its performance gain over legacy clients. We do so by isolating its key ingredients to understand what is their contribution to the improved performance. We then make the case for an extreme scenario in which all users would adopt BitTyrant and discuss its implications on the whole community.

The main contributions of this work can be summarized as follows:

- We generalize the analytical model presented in [15] to identify the extent to which BitTorrent can be exploited by greedy clients. Unlike previous results discussed in

[15], our findings indicate that exploiting the altruism of BitTorrent is effective only during a *short* transient regime when the system is bootstrapping;

- We study the different components of a prominent example of a greedy client, BitTyrant [15], and we evaluate to what extent each part of the proposed solution is responsible for the performance achieved; we also compare the results with the ones obtained by the mainline BitTorrent client;
- We cast light on the subtle choke algorithm used by BitTyrant and show its unexpectedly positive impact on system performance, especially during the most delicate phase of content distribution, the startup phase;
- Finally, we make the case for a wide adoption of the BitTyrant client by the mass. We show that the interaction of BitTyrant clients may lead to an undesirable state with some peers progressively throttling the uploading rate to their neighbors and others intermittently choking their contribution, resulting in poor system performance.

The remainder of the paper is organized as follows: Sec. 2 provides some background on BitTorrent and its variants; in Sec. 3 we analyze to which extent BitTorrent can be exploited by greedy strategies; Sec. 4 presents a simulation-based performance evaluation of a single BitTyrant client and pinpoints the merit of its key components to the increased performance; finally, in Sec. 5 we make the case of a wide-spread adoption of BitTyrant and analyze the implications on global system performance.

## 2 Background

In this section we briefly outline the key algorithms used by BitTorrent [1], BitTyrant [15] and BitThief [14].

**BitTorrent.** The BitTorrent protocol is designed for bulk data transfer. The file is divided into pieces, which can be downloaded in parallel from peers belonging to a specific *torrent*. A central entity, called *tracker*, keeps track of all peers downloading the content and bootstraps new peers joining the torrent with a random set (of size 50 peers) of remote peers to connect to: the neighborhood of a peer is called the **peer set**.

A BT peer executes two key algorithms, one that is used to select pieces of the content to download (termed the piece selection algorithm) and one that is used to select remote peers to upload data to (termed the peer selection algorithm, or the *choke algorithm*). In this work we focus on

the choke algorithm, and gloss over the details of piece selection. With the choke algorithm, a node builds a subset of its peer set that is termed **active set**: peers in the active set are entitled to request pieces of the content. The choke algorithm is executed every 10 seconds: all remote peers are ranked based on their upload rate and only the first  $k$  top peers are unchoked. Along with regular unchokes, every 30 seconds a peer randomly unchokes  $\omega$  peers irrespectively of their rank: this technique is termed optimistic unchoke and allows a peer to explore its peer set and discover fast neighbors. With the choke algorithm, peers discover and maintain an active set (of size  $k + \omega$ ) composed by neighbors that maximize *reciprocation*, i.e. the amount of data downloaded given the amount of data uploaded to remote peers.

In the basic version of BT,  $k$  and  $\omega$  are empirically set parameters: generally  $k = 4$  and  $\omega = 1$ . This configuration is used also by Azureus. The upload bandwidth of a peer is shared equally (beside TCP effects) among all unchoked peers; the portion of the bandwidth that each peer is able to obtain is defined as **equal-split**.

Recently, a new version the mainline BT protocol has been released. Despite its rather small diffusion among users (only 2% of the clients appear to be of type mainline [15]), we analyze in this work the impact of this new client, that we termed **BTnew**. The key difference of BTnew lies in the choice of the parameters of the choke algorithm. The number of regular unchokes is determined as a function of the uplink capacity  $C$  of a peer, that is  $k = \sqrt{0.6C}$  ( $C$  is expressed in KBytes/s). Moreover,  $\omega = 2$ . With these new parameters, peers with a high uplink capacity open more active connections.

**BitTyrant.** The key modifications introduced by BitTyrant (hereinafter **BTyr**) are related to the peer selection algorithm. As for BTnew, the number of unchoked peers is a function of a peer's uplink capacity. However, BTyr uses a dynamic bandwidth allocation algorithm by which uplink capacity is assigned on a per-connection basis. During the initial phase of the download process, a BTyr peer allocates the same bandwidth  $c = 15$  KBytes/s to all connections. This initial value, found empirically, is set such that the probability of reciprocation from remote peers is high. The authors in [15] compute  $c$  considering a bandwidth distribution derived from real measurements: as in this work we adopt the same distribution, we also use the same value for  $c$ .

Subsequently, the alternative choke algorithm works as follows: if a remote peer reciprocates for at least 3 unchoking intervals, the bandwidth allocated for this active connection is reduced by a factor of 0.9. If an unchoked neighbor stops reciprocating, then the bandwidth allocated to the active connection is increased by a factor 1.2. Every choke in-

terval (set to 10 sec.), neighbors are sorted according to the ratio between the amount of data *received* and *sent* (in the last 20 sec.); the available uplink capacity is then progressively allocated to remote peers in descending order. Hence, the amount of bandwidth allocated to a remote peer should converge to the exact value required to guarantee reciproca-tion.

**BitThief.** The primary aim of this client was to show the intrinsic weakness of the optimistic unchoke adopted by BT. BitThief continues to contact the tracker in order to increase as much as possible its peer set size. As a consequence, the probability to be optimistically unchoked increases, and the client can receive the content without uploading at all.

### 3 Misuse Opportunities in BitTorrent: an Analytical Perspective

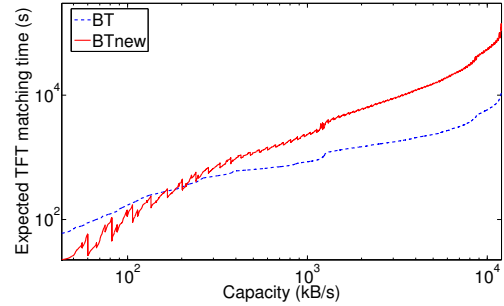
In this section we take a data agnostic approach and analyze the extent to which the altruistic behavior of both BT and BTnew might be exploited by self-interested peers. Our analysis extends and formalize rigorously the key observations made in [15], which are behind the design of BTyr. We do not consider here the BitThief scheme since the evaluation of its benefits are straightforward.

#### 3.1 Matching Time

As noted in prior studies [16, 17], the choke algorithm, which constitutes the basis of the peer selection process, can be seen as a distributed algorithm for the stable  $b$ -matching problem, that converges to a (weakly) stable state in which peers are matched based on their upload capacity and no peer has an incentive to deviate from its matches. The algorithm converges to a stable state through a series of exploration rounds (i.e. optimistic unchokes) in which unstable matchings are formed: in such intermediate cases, a peer may end up being matched to remote peers that cannot sustain a fair reciprocation. This implies that some peers might offer more upload bandwidth than they receive.

The time it takes for the algorithm to converge could be exploited by a peer striving for maximizing the reciprocation it receives from remote peers. In the following we endeavor to quantify the convergence time, termed *matching time* hereinafter. The matching time we derive ignores (i) the peer churn rate, (ii) the content availability and (iii) that some remote peers could be not willing to reciprocate. The last issue is going to be taken into account in the following section.

During a time interval equal to  $T_{opt}$ , a peer discovers (using optimistic unchokes) the equal split of  $\omega$  new peers and its equal split is discovered by other  $\omega$  new peers. Given



**Figure 1. Time required for a new peer to discover a number of peers of equal or greater equal-split to fill its active set.**

peer  $i$  with equal split  $u_i$ , let  $A_i$  be the set of active connections (neighbors it has unchoked). We denote with  $b(u)$  and  $B(u)$  respectively the Probability Density Function (PDF) and the cumulative distribution function (CDF) of the equal split.  $b(u)$  ( $B(u)$ ) can be evaluated through an empirical distribution<sup>1</sup>.

The expected number of interactions peer  $i$  needs to find an attractive peer is geometrically distributed, with expected value  $1/(1 - B(u_i))$ . The expected number of interactions needed to discover a number of peers equal to the number of active connections  $|A_i|$  is simply  $|A_i|/(1 - B(u_i))$ . If we consider that the peer has one interaction every  $T_{opt}/(2\omega)$  seconds, then the matching time is:

$$\frac{T_{opt}}{2\omega} \frac{|A_i|}{1 - B(u_i)}. \quad (1)$$

The equation shows that the matching time increases when the number of active connections or the equal split increases.

In Fig. 1 we show the matching time for BT and BTnew clients with different uploading capacities. Matching times are as large as 1 and 10 hours respectively for high capacity BT and BTnew clients. The sawtooth behavior of the BTnew curve is due to non-monotonic relation between the uploading capacity and the equal split. Given two peers with similar capacities, it can happen that the one with higher capacity opens one additional connection; in this case, its equal-split is smaller and the time needed to discover faster peers is lower.

Long matching times paves the way for clients such as BTyr that tries to exploit high-capacity peers as long as their discovery phase has not converged yet.

<sup>1</sup>In this work we use the same empirical distribution as in [15].

### 3.2 Probability of Reciprocation and Expected Download Rate

The extremely long convergence time (especially with respect to typical download times) toward a stable matching that we discussed in the previous section has encouraged the design of subtle techniques [15] to exploit peers until a global matching is reached. By that time, peers would be immune to greedy strategies. A greedy peer, however, is not guaranteed to be reciprocated from remote peers at all times during the matching time.

We show this by studying the *evolution in time* of the probability of reciprocation and its impact on the expected download rate of a peer. The following analysis constitutes a significant extension to that sketched in [15]. As noted above, the download rate peer  $i$  can achieve varies over time. Indeed peer  $i$  can select its  $|A_i|$  best uploaders from a progressively larger set, but reciprocation from its peer set fluctuates: reciprocation from peers with higher capacity decreases (because they discover similar peers), while reciprocation from lower capacity peers increases (because they are progressively choked by their best uploaders). Being that each peer optimistically unchokes  $\omega$  new peers every  $T_{opt}$ , we consider a discrete time system where every  $T_{opt}/(2\omega)$  seconds each peer discovers the equal split of a new peer. Let us define  $\rho(u_i, u_j, k)$  the probability that a node with equal split  $u_j$  is willing to reciprocate with a node with equal split  $u_i$  at the  $k$ -th interaction. The probability that a generic peer is willing to reciprocate to peer  $j$  at the  $k$ -th interaction is

$$\int_0^\infty \rho(u_j, v, k) b(v) dv,$$

and the expected number of peers not reciprocating peer  $j$  ( $\bar{R}_j(k)$ ) is:

$$\bar{R}_j(k) = k \left( 1 - \int_0^\infty \rho(u_j, v, k) b(v) dv \right).$$

We simplify our analysis assuming that: (i) the number of peers not reciprocating peer  $j$  is always equal to the integer nearest to  $\bar{R}_j$  (we denote it as  $\hat{R}_j$ ) and (ii) that these peers are the best uploaders of peer  $j$ . Then if we rank the uploaders of peer  $j$  on the basis of their equal split in decreasing order, peer  $j$  at the  $k$ -interaction will be willing to reciprocate peers with rank from  $\hat{R}_j(k) + 1$  to  $w_j = \hat{R}_j(k) + |A_j|$ , assuming that it is willing to open up to  $|A_j|$  connections. Now the probability that peer  $i$  is going to be reciprocated from peer  $j$  at the following interaction is equal to the probability that peer  $i$  has an higher equal split than that of the  $w_j$ -th uploader of peer  $j$ <sup>2</sup>. We can then

<sup>2</sup>If  $w_j = \hat{R}_j(k) + |A_j| > k$ , peer  $j$  will be always willing to reciprocate with a new peer.

use order statistic results to derive the equal split PDF of the  $z$ -th uploader of peer  $j$ :

$$b_{u_j}^{(z)}(v, k) = \frac{k!}{(z-1)!(k-z)!} B(v)^{k-z} (1-B(v))^{z-1} b(v).$$

The reciprocation probability at the  $k+1$ -th iteration can be evaluated considering that peer  $i$  will be reciprocated by peer  $j$  only if it will be better than the  $w_j$ -th best uploader of peer  $j$ , then:

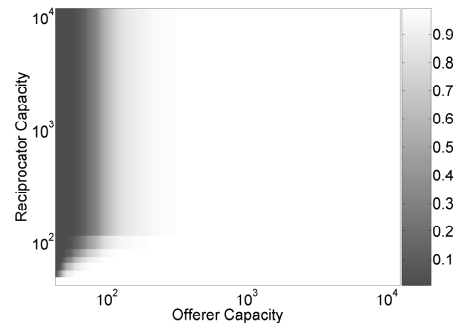
$$\rho(u_i, u_j, k+1) = \int_0^{u_i} b_{u_j}^{(w_j)}(v, k) dv. \quad (2)$$

The system starts from a state where every peer has an empty active set and it is willing to reciprocate with everyone else ( $\rho(u_i, u_j, 0) = 1$ ), then Eq. 2 can be used to evaluate the evolution of reciprocation probabilities.

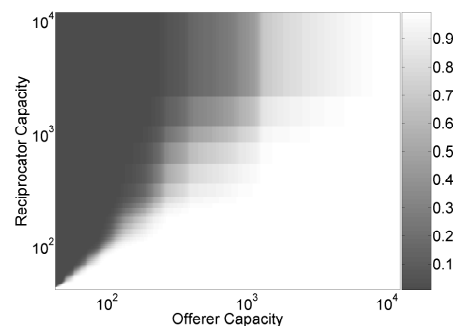
The expected download rate of peer  $j$  can be derived as:

$$\sum_{h=\hat{R}_j(k)+1}^{\hat{R}_j(k)+|A_j|} \int_0^\infty v b_{u_j}^{(h)}(v, k) dv + \omega \int_0^\infty v b(v) dv, \quad (3)$$

where the first term corresponds to the aggregated rate from active connections, while the second one to the aggregated rate from optimistic unchoking.



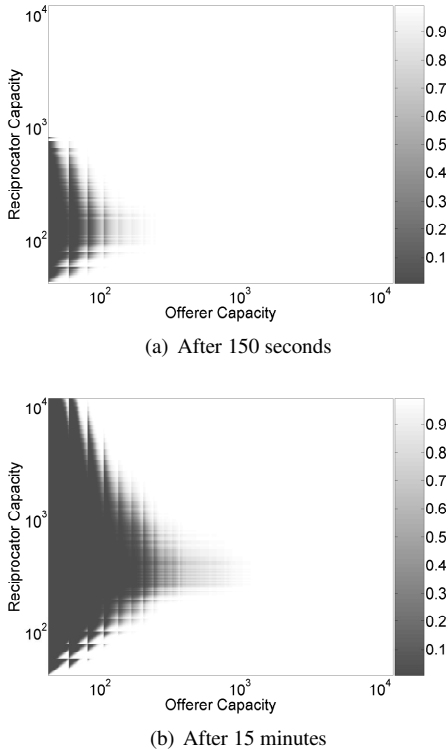
(a) After 150 seconds



(b) After 15 minutes

**Figure 2. Reciprocation probability for BT.**

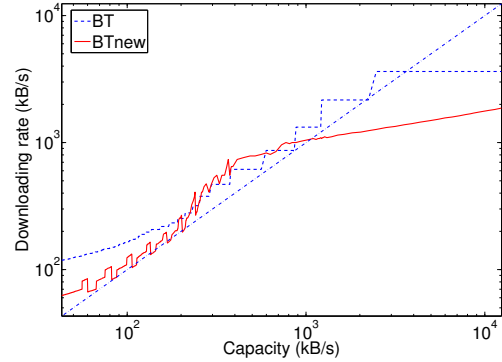
Fig. 2 shows the reciprocation probability for BT clients after 150 seconds and after 15 minutes, respectively the time intervals needed by each peer to discover the equal splits of 10 and 60 peers. Every point  $(x, y)$  of the figure indicates the probability that a peer with capacity  $x$  is going to be reciprocated by a peer with capacity  $y$ . After 150 seconds (Fig. 2-a), peers with lower uplink capacities are very unlikely to be reciprocated by fast peers; however, the probability for fast peers to reciprocate remote peers that cannot sustain their upload rates is very close to one. This observation no longer holds after 15 minutes, (Fig. 2-b): in this case a large fraction of peers is willing to reciprocate only with other peers with similar or higher capacities.



**Figure 3. Reciprocation probability for BTnew.**

Fig. 3 shows the corresponding results for the BTnew client. BTnew appears to be more generous in that the probability of an unfair reciprocation (a slow peer being served by a fast one) is still high after 15 minutes.

Fig. 4 reports the expected download rate of a peer with a given uplink capacity, after 15 minutes from the beginning of the download process. Fairness is achieved when the uplink capacity equals the expected download rate (diagonal line in the figure). We recall that both regular and optimistic unchokes contribute to the download rate observed by a peer. In the BT case, Fig. 4 illustrates that low capac-



**Figure 4. Expected download rate for a peer of a given capacity after 15 minutes.**

ity peers are able to get more than their fair rate. This is mainly due to optimistic unchokes: focusing only on regular unchokes would reveal that the expected download rate is parallel to the diagonal up to roughly 200kB/s. On the contrary, peers with capacity greater than 3000kB/s offer more upload capacity than they receive: this is exploited by peers with intermediate upload capacity.

While similar observations can be drawn for the BTnew case, we notice that the advantage for low capacity peers is less pronounced: this is due to the larger number of active connections (hence lower uplink bandwidth dedicated to each of them) of a BTnew client.

### 3.3 Discussion

Our data agnostic analysis indicates that exploiting BT or BTnew clients appears tempting in a first instance, if one considers the time required by the peer selection to stabilize. However, due to the variability in time of the probability of reciprocation, a greedy strategy would work best during the initial stages of the download process, where high capacity peers are still willing to serve low and intermediate capacity peers.

This conclusion raises the legitimate question of whether these results carry over when *piece availability* is considered. Indeed, piece availability plays a crucial role, especially during the initial phase of the download process, when the number of pieces being exchanged by peers is scarce. This key observation calls for a deeper study of the performance that can be achieved by a greedy client. Due to the complexity of the analysis when piece availability is taken into account, we revert in the following to a simulation-based performance analysis.

## 4 Deconstructing BitTyrant: the Single Client Case

In the following we carry out a simulation-based analysis of the performance of a prominent example of a strategic client, BTyr. We decided to focus on BTyr because it merges several greedy techniques previously discussed in the literature [14, 13]: (i) greedy peer set size and (ii) greedy uplink allocation :

- (i) implies that peer set size in BTyr is larger than that of a traditional BT or BTnew client (this approach is adopted also in BitThief [14]); the consequence is that the probability of being optimistically unchoked is higher;
- (ii) implies that the uplink capacity of a peer is not equally split among its active connections, but shaped according to a greedy objective; hence, the number of active connections is not a fixed parameter but varies over time.

Here we deconstruct the BTyr client to understand the contributions of its building blocks to the increased performance achieved by a **single** BTyr client in a torrent of BT or BTnew clients.

### 4.1 Simulator Description, Methodology and Settings

Our work is based on a customized version of the publicly available BitTorrent simulator called GPS [18]. GPS is a discrete time flow level simulator, featuring a simple fluid model of TCP: the available bandwidth between two peers is equally shared among active flows on the path joining the peers. Peers have infinite downlink capacity and a finite uplink capacity, which is distributed according to the bandwidth distribution of [15]. It implements the BT client, including the piece selection, the choke algorithm and the tracker. We complemented the simulator with an implementation of (i) the new version of the *mainline* BT client (BTnew) and (ii) the BTyr client.

The main performance metrics we use are:

**Download time** of the single client (BT, BTyr or BTnew) in the different scenarios (all BT and all BTnew);

**Number of pieces** uploaded by the single client during the download process;

**Empirical Cumulative Distribution Function** (ECDF or CDF) of the download time of all peers.

For the BTyr case, we also characterize the uplink capacities of the peers unchoked over time. Note that when we focus

on a single client, we compare the performance of one peer using BTyr, BT or BTnew clients in the same simulation conditions.

We analyze torrents of 350 peers where one initial seed distributes a file of 50 MB. We select this file size since the gain of a strategic client is mainly concentrated at the beginning of the distribution process (as showed in the analysis in Sect. 3), thus BTyr should benefit more from short torrents than larger ones. Peers randomly start to download the content within a small interval of time (10 sec.) and stay as seeds in the system once they finish downloading the content. For each scenario, we perform 10 simulation runs, generating different random arrival patterns, where peers have different bandwidths, randomly selected from the bandwidth distribution. We estimate the mean download time, along with the confidence interval for a confidence level of 95%.

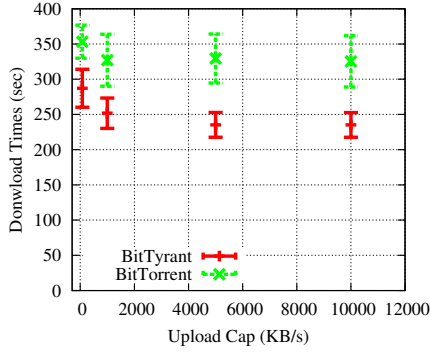
### 4.2 Impact of the Peer Set and Active Set Size

In this Section we build a *baseline* scenario in which a single, fully-fledged BTyr client operates in a torrent of BT or BTnew peers. We then artificially obstruct the greedy peer set construction of BTyr by limiting the frequency of requests to the tracker: the peer set size is then equal at most to 80 for every peer in the torrent.

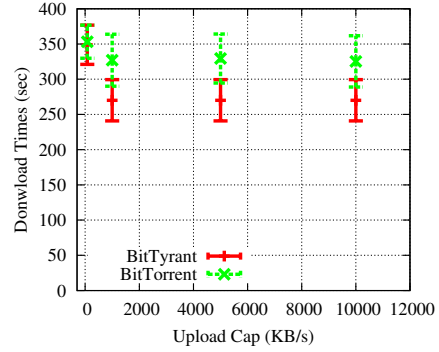
Fig. 5(a) illustrates the download time of a single BT, and BTyr client for different classes of uplink capacity in the *baseline* case. Similarly, Fig. 5(b) depicts the download time of a BTnew client versus a BTyr client. We observe that the performance gain of BTyr over BTnew dramatically drops as compared to the same setting when BT is used. This is due to the large number of active connections established by fast peers using BTnew. Their uplink capacity is over-carved, hence remote peers receive smaller download rates as compared to the original BT algorithm.

Figs. 6(a) and 6(b) show the download time for the same set of experiments shown above when the greedy peer set construction is obstructed. The results illustrate a significant performance loss of BTyr in a torrent of both BT and BTnew clients, indicating that the increased peer set size constitutes one of the *main* factors influencing download performance.

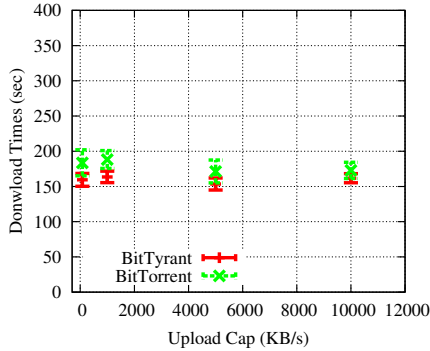
We note that, with BTnew clients, BTyr could completely lose all its benefits. BTyr not only uses a larger peer set size, but also a larger active set size, i.e. it maintains many active connections, giving a small fraction of bandwidth to each of them. Assuming that this policy provides a gain (we will show in Sect. 4.3 *why* it actually does), in an environment where other peers use the same approach – i.e. in a torrent with all BTnew clients – the benefits of maintaining many active connections should be limited. The re-



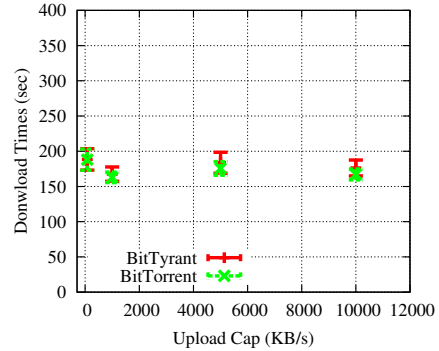
(a) BTyr with all BT



(a) BTyr with all BT



(b) BTyr with all BTnew



(b) BTyr with all BTnew

**Figure 5. Mean download time of a single client with different bandwidths.**

**Figure 6. Mean download time of a single client with a constrained peer set.**

sults shown in Figs. 6(b) confirm this observation. As a further test, we have considered a single BTnew client that operates in a torrent of BT clients. Our experiments show that a single BTnew client achieves similar performance as BTyr (see Fig. 6(a)).

These results hint at the fact that the dynamic uplink bandwidth allocation algorithm adopted by BTyr appears to have little impact on performance. We further note that our simulation study reveals to be necessary: piece availability plays a crucial role that could not be understood using a simplified theoretic formulation of the problem.

### 4.3 Impact of Greedy Uplink Capacity Allocation

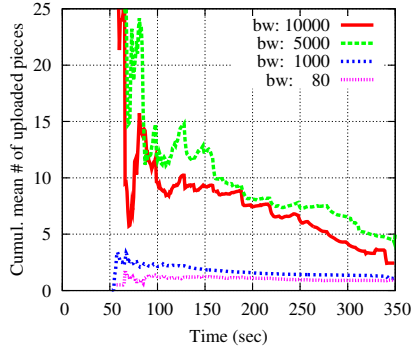
In the previous section we unveiled that the performance gain of BTyr is mainly due to the larger peer set and active set. While the effect of a larger peer set is well understood, we discuss here the advantage of a larger active set, along with the impact of the subtle uplink bandwidth allocation strategy of BTyr.

The rationale behind the BTyr design is to dynamically adapt both the uplink capacity dedicated to a remote un-

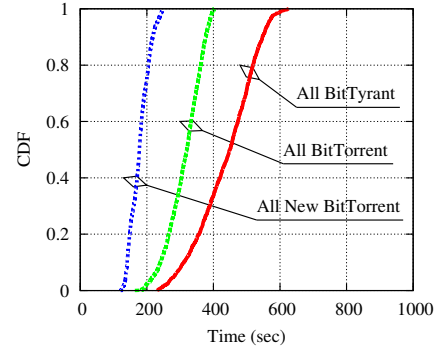
choked peer and the number of active connections (i.e. unchoked peers) so as to maximize the probability of reciprocation. Our analysis showed that this technique can be exploited best during the initial stages of the download process, which however is characterized by low piece availability.

On the one hand, by keeping a larger number of active connections, BTyr strives for maximizing the chance of being reciprocated, with the knowledge that reciprocation will happen on a tit-for-tat basis due to the choke algorithm. On the other hand, since during the initial phase of the download process the lack of fresh pieces to serve could cause uplink capacity underutilization, a larger active set size helps spreading available pieces to a large number of peers that would otherwise remain unserved. This increases the utilization of the uplink capacity of both the BTyr peer and its neighbors. Interestingly, the greedy strategy adopted by BTyr has actually a hidden altruistic nature.

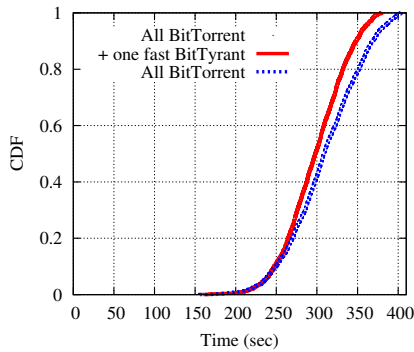
Fig. 7 depicts the ratio between the cumulative number of uploaded pieces over time by single BTyr client with respect to the corresponding BT client. Especially during the early stages of content distribution, BTyr uploads up to



**Figure 7. Time series of the ratio between cumulative uploaded pieces by BTyr and BT.**



**Figure 9. CDF of download times for BT, BTnew and BTyr.**



**Figure 8. CDF of download times with or without a single standard BTyr client with bandwidth 5000 KB/s.**

25 times (for a high bandwidth peer) the number of pieces uploaded by BT. During steady state, the total number of pieces uploaded for BTyr and BT converges.

The unexpected altruism of BTyr has a beneficial effect on all peers involved in the distribution process. In Fig. 8 we show the cumulative distribution function (CDF) of the download times of all peers in the system. Results indicate that, when even only one fast peer (with high bandwidth equal to 5000 KB/s) adopts BTyr instead of BT, there is a positive impact on the performance of all the other peers. We also note that similar observations can be made when introducing one BTnew client.

The results obtained in this section hint toward an important direction of future research, that is the study of dynamic uplink allocation algorithms, where the number of active connections is not an empirically set parameter as done in BT. However, we show next that the apparently attractive uplink allocation strategy of BTyr cannot readily be used by all peers in a system.

## 5 The Multiple Clients Case

The results presented in the previous section indicate the potential performance improvement of a greedy client such as BTyr, and its counter-intuitive positive impact on a torrent. Authors in [15] make the point that there are reasons to assume an increasing popularity of BTyr and present some initial results for the case of a torrent of all BTyr clients. They argue that a wide-spread adoption of BTyr may have a negative impact on global system performance. To cope with this problem, [15] suggest the following fix: when peers establish a connection and perform the initial handshake, if they realize that they both are using BTyr, they should switch to a block based TFT strategy. There are however no hints toward any incentive compatibility of this approach: truthful revelation (revealing that a peer is using BTyr) may not be a dominant strategy<sup>3</sup>.

In this section we take a different perspective and progressively isolate the effects of the peculiar uplink allocation strategy of BTyr to understand exactly why system performance degrade when all peers use it. First, we analyze system performance when all clients are BTyr and they use a peer set size at most equal to 80. Note that this approach also reflects a recent trend of commonly deployed trackers that implement some sort of access control mechanism to limit the frequency of the requests from peers greedily trying to extend their peer set size.

Fig. 9 illustrates the CDF of the download times for a torrent of all BT, BTnew and BTyr clients: a glance at the median and worst case download times indicates that a large-scale adoption of BTyr can indeed jeopardize the content distribution process, even with a constrained peer set size.

<sup>3</sup>If peer  $i$  uses BTyr and lies it may be better off when facing an honest peer, or worse off if facing another liar. It is out of the scope of the paper to analyze this simple game.

In contrast, the best performance is achieved by a torrent formed by BTnew clients only.

We now deepen our analysis and neglect the effect of piece availability in our simulations. We assume that each peer has always interesting pieces to serve, hence a peer’s uplink capacity can be always fully utilized. This approach allows to focus only on the exact values allocated by the BTyr choke algorithm to remote peers, rather than on the actual amount of data sent or received.

Next, we show the uplink rate assigned by a peer to each neighbor, over time: for every peer  $k$  in the system we maintain a matrix  $E^{(k)}$  where the element  $e_{ij}^{(k)}$  represents the rate assigned by peer  $k$  to peer  $i$  at choking interval  $j$ . Fig. 10 illustrates  $E^{(k)}$  for a peer  $k$  with 10000 KB/s uplink capacity. The value of rate is visualized using shades of gray: the darker regions indicate higher rates. During the initial phase of the download process, peer  $k$  allocates the same uplink rate to all its neighbors<sup>4</sup>.

The uplink capacity allocated by peer  $k$  varies over time, and it’s possible to observe two different trends: (i) some neighbors of peer  $k$  are allocated less and less uplink bandwidth; (ii) other neighbors are assigned an increasing amount of bandwidth, which then degenerates into a periodic, on-off, phase.

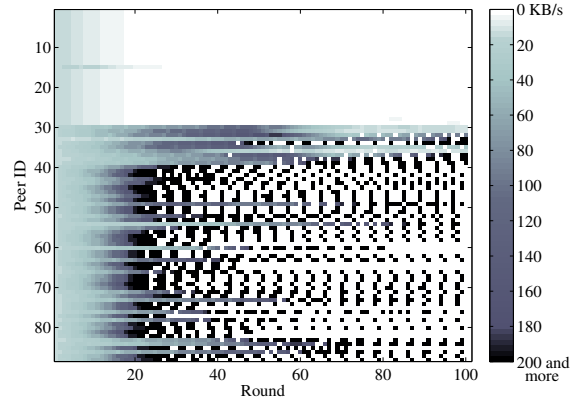
We now focus on the latter case. The initial increasing trend can be explained as follows: on the one hand, peer  $k$  has *spare* uplink capacity, thus it unchokes *all* its neighbors; on the other hand, its neighbors may have limited capacity, hence they choke peer  $k$ . As a consequence, peer  $k$  (that follows the BTyr choke algorithm) increases the uplink capacity to remote peers to increase the probability of reciprocation. This behavior is visible for the first 20-30 rounds. At this point, the rate allocated by peer  $k$  to remote peers reaches a very high value. As a consequence, (i) peer  $k$  starts choking some neighbors, since it does not have enough capacity for all of them; (ii) on the contrary, peer  $k$ ’s neighbors start unchoking it. These two phases are interleaved and concur in creating the periodic behavior.

A close look at Fig. 10 indicates that the periodicity is equal to three rounds<sup>5</sup>. This is a consequence of the probing period used by BTyr (and BT) to estimate the received/sent rate.

The instability we emphasize here clearly arises due to an implicit feedback loop that is created when two peers interact. However, the uplink allocation strategy of BTyr cannot handle this situation, which may appear in the case of a wide-spread adoption of this modified client. In the extreme case of a torrent composed by BTyr clients only, the whole content distribution process may be disrupted.

<sup>4</sup>Note that, since the initial rate for each unchoked neighbor is 15 KB/s, peer  $k$  unchokes *all* its neighbors; moreover, the total amount of assigned capacity is in any case lower than the available uplink capacity.

<sup>5</sup>Similar results are obtained for a peer  $k$  with different uplink capacity.



**Figure 10. Upload rate in the multiple clients case: snapshot for a fast BTyr client.**

## 6 Conclusion

Recent days have witnessed the development of new, greedy peer-to-peer clients aiming at decreasing content download times by leveraging on subtle techniques to exploit generous clients. In this work we focused on BitTorrent networks and analyzed two commonly deployed greedy techniques (implemented in BitTyrant), while we glossed over explicit misbehaviors such as pollution attacks. We showed that the BT protocol can be misused to gain an advantage over standard peers by building progressively larger peer sets; we noted however that it is straightforward to protect against such a greedy technique.

We then argued that further work on more sophisticated choke algorithms would constitute an important avenue for future research. Indeed, we showed the greedy uplink allocation algorithm of BitTyrant has some unexpected positive implications on the content distribution process, especially during its bootstrap phase. However, our results indicated that this greedy algorithm could not be readily deployed in a setting in which multiple (if not only) greedy client would coexist.

Along the same lines, our results on the last version of the legacy BitTorrent client suggested a significant performance improvement due to a larger number of active connections for high capacity peers.

As part of our current research agenda, we are studying new uplink allocation algorithms that, if adopted by the majority of clients in a BitTorrent network, would radically boost its performance.

## Acknowledgements

This work has been partially funded by the Integrated Project CASCADAS (FET Proactive Initiative, IST-2004-

2.3.4 Situated and Autonomic Communications) within the 6th IST Framework Program.

## References

- [1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of P2P-Econ*, Berkeley, California, USA, June 2003.
- [2] <http://www.guardian.co.uk/technology/2006/oct/19/guardianweeklytechnologysection.insideit>
- [3] <http://www.theglobeandmail.com/servlet/story/RTGAM.20071128.wgtbittorrent29/BNStory/Technology>
- [4] <http://arstechnica.com/news.ars/post/20080421-study-bittorrent-sees-big-growth-limewire-still-1-p2p-app.html>
- [5] A. Legout and G. Urvoy-Keller and P. Michiardi "Rarest first and choke algorithms are enough," in *Proc. of IMC*, Rio de Janeiro, Brazil, October 2006.
- [6] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems", in *Proc. of IMC*, Berkeley, California, USA, October 2005.
- [7] M. Izal, Guillaume Urvoy-Keller, Ernst W. Biersack, Pascal Felber, Anwar Al Hamra, Luis Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime" in *Proc. of PAM*, Antibes, France, April 2004.
- [8] X. Yang and G. de Veciana, "Performance of Peer-to-Peer Networks: Service Capacity and Role of Resource Sharing Policies," in *Performance Evaluation*, Vol. 63, Issue. 3, 175-194, March 2006.
- [9] A. R. Bharambe, C. Herley, V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," in *Proc. of INFOCOM*, Barcelona, Spain, May 2006.
- [10] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. of SIGCOMM*, Portland, Oregon, USA, August 2004.
- [11] F. Bin, D. M. Chiu, J. C.S. Lui, "The Delicate Trade-offs in BitTorrent-like File Sharing Protocol Design," in *Proc. of ICNP*, Santa Barbara, USA, 2006.
- [12] F. Bin, D. M. Chiu, J. C.S. Lui, "Stochastic Differential Equation Approach to Model BitTorrent-like P2P Systems," in *Proc. of ICC*, Istanbul, Turkey, June 2006.
- [13] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting BitTorrent For Fun (But Not Profit)," in *Proc. of IPTPS*, Santa Barbara, California, USA, February 2006.
- [14] T. Locher and P. Moor and S. Schmid and R. Wattenhofer "Free Riding in BitTorrent is Cheap," in *Proc. of HotNets-V*, Irvine, California, USA, November 2006.
- [15] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in BitTorrent?," in *Proc. of NSDI*, Cambridge, MA, USA, Apr. 2007.
- [16] A. Legout, N. Liogkas, E. Kohler, and L. Zhang "Clustering and Sharing Incentives in BitTorrent Systems," in *Proc. of SIGMETRICS*, San Diego, CA, USA, June 2007.
- [17] A. Gai, F. Mathieu, F. de Montgolfier, J. Reynier, "Stratification in P2P networks: Application to BitTorrent," in *Proc. of ICDCS*, Toronto, Ontario, Canada, July 2007.
- [18] W. Yang and N. Abu-Ghazaleh, "GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent," in *Proc. of MASCOTS*, Atlanta, Georgia, USA, September 2005.