

# On the Robustness of BitTorrent Swarms to Greedy Peers

Damiano Carra, Giovanni Neglia, *Member, IEEE*,  
Pietro Michiardi, *Member, IEEE*, and Francesco Albanese

**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 to 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 possible 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 analyze in detail.

**Index Terms**—Peer-to-peer, BitTorrent, performance analysis.

## 1 INTRODUCTION

BITTORRENT [1] is a peer-to-peer (p2p) content distribution application that has been adopted by millions of end users [2], [3], [4]. BitTorrent (BT) has also attracted the attention of a large body of researchers that focused on its building blocks (such as the incentives, as in [5]) and its performance analysis through measurement [6], [7], [8], simulation [9], [10], and analytical [11], [12], [13] studies. These previous works indicated that the key of BitTorrent success can be substantially attributed to its scalability and its greater robustness to free riding in comparison to previous p2p proposals.

Some recent studies [14], [15], [16] have proposed new clients, that are compliant to BitTorrent message protocol, but adopt greedy strategies with the purpose of optimizing the local performance of the modified client. For example, Locher et al. [15] designed the BitThief client, that tries to maximize the content download rate without uploading any content. Another prominent example is that of BitTyrant [16], which tries to maximize its download rate by shaping its contribution to remote peers. Note that while BitThief client is intrinsically a free rider, BitTyrant makes its whole upload capacity available to spread the content.

Our research interest is twofold. First, we want to evaluate the competitive advantage of greedy clients over standard

ones and hence the possibility of wide adoption by the peer-to-peer community. Second, we want to investigate the possible effects that a widespread adoption could induce to the system performance. 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 standard clients. We do so by isolating its key mechanisms to quantify their contribution to the improved performance. Some of these results were presented in a preliminary form in [17]. We then make the case for a gradual adoption of BitTyrant by users 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 [16] to identify the extent to which BitTorrent can be exploited by greedy clients. Unlike previous results discussed in [16], 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 [16], and we evaluate to what extent each part of the proposed approach 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: while we show its unexpectedly positive impact on system performance—especially during the startup phase of content distribution—we also point to an undesired periodic behavior that limits its performance;
- finally, we make the case for a gradual adoption of the BitTyrant client by the mass; we show that a widespread adoption of BitTyrant clients seems

- D. Carra is with the Department of Computer Science, University of Verona, Strada le Grazie 15, Verona I-37134, Italy.  
E-mail: damiano.carra@univr.it.
- G. Neglia is with the INRIA, 2004 Route des Lucioles, BP 93, Sophia Antipolis F-06902, France. E-mail: giovanni.neglia@ieee.org.
- P. Michiardi and F. Albanese are with the EURECOM, 2229 route des Cretes, BP 193, Sophia Antipolis F-06560, France.  
E-mail: {michiard, albanese}@eurecom.fr.

Manuscript received 2 July 2009; revised 30 Dec. 2009; accepted 5 Nov. 2010; published online 14 Mar. 2011.

Recommended for acceptance by J. Cao.

For information on obtaining reprints of this article, please send e-mail to: [tpds@computer.org](mailto:tpds@computer.org), and reference IEEECS Log Number TPDS-2009-07-0295. Digital Object Identifier no. 10.1109/TPDS.2011.94.

unrealistic, and in any case the increasing adoption of BitTyrant progressively degrade the system performance.

## 2 BACKGROUND

In this section, we briefly outline the key algorithms used by BitTorrent [1], BitTyrant [16], and BitThief [15].

### 2.1 BitTorrent

The BT protocol is designed for the bulk data transfer. A file is divided into pieces, which can be downloaded in parallel from peers belonging to a specific *torrent*. A central entity, the *tracker*, keeps track of all peers sharing the content and provides new peers a random set of 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 (piece selection algorithm) and one that is used to select remote peers to upload data to (the *choke algorithm*). In this work we focus on the choke algorithm. With the choke algorithm, a node builds a subset of its peer set that is termed the *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. The choke algorithm aims at maintaining a high level of *reciprocation* among peers, being that peers uploading less have less chances to be unchoked by their neighbors.

In the first version of BT—that we term **BTold**— $k$  and  $\omega$  are empirically set parameters: generally  $k = 4$  and  $\omega = 1$ . 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*. In the latest version of the BT mainline protocol the choice of the parameters in the choke algorithm is different: the number of regular unchokes is determined as a function of the uplink capacity  $C$  of a peer, that is  $k = \lfloor \sqrt{0.6 \cdot C} \rfloor$  ( $C$  is expressed in KBytes/s). Moreover,  $\omega = 2$ . With these new parameters, peers with a high uplink capacity open more active connections.

Note that the values of the parameters  $k$  and  $\omega$  in BT and BTold are completely arbitrary; the interested reader can find a discussion about the impact of these parameters in [18].

In rest of the paper, BT is the main reference to evaluate the performance of greedy clients, whereas in the online extension of this work we include the results we obtained focusing on BTold.

### 2.2 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.

### 2.3 BitTyrant

BitTyrant (hereinafter BTyr) adopts the same mechanisms of BitThief, but also introduces a new peer selection

algorithm. As for BT, 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 has been derived in [16] from a measured peer bandwidth distribution in order to guarantee that the probability of reciprocation (i.e., the probability of being unchoked) from remote peers is high.

Subsequently, the alternative BTyr choke algorithm works as follows: if a remote peer reciprocates for at least three 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 interval (set to 10 seconds), neighbors are sorted according to the ratio of the amount of data *received* to the amount of data *sent* in the last 20 seconds; 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 reciprocation.

## 3 MISUSE OPPORTUNITIES IN BITTORRENT: AN ANALYTICAL PERSPECTIVE

In this section, we analyze the extent to which the altruistic behavior of BT might be exploited by self-interested peers. We adopt a *data agnostic* approach, i.e., we assume that each peer has always pieces that every other is missing. The analysis extends and formalize rigorously the key observations made in [16], which are behind the design of BTyr.

Note that the analysis is focused on the possible exploitation of BT: how a client is going to use this information is independent from the analysis. In the rest of the paper, we will focus only on BTyr as strategic client, because it merges several greedy techniques discussed in the literature. 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 [19], [20], the choke algorithm 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. During this intermediate phase a peer may be 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 derive a rough estimation of the convergence time, termed *matching time* hereinafter. Our analysis of the matching time: 1) assumes a large swarm, with a fixed peer population, 2) does not take into account content availability, and 3) ignores that some remote peers could be not willing to reciprocate. The last issue is going to be addressed 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 peer  $i$  with equal split  $u_i$ , let  $A_i$  be the set of its active connections (i.e., the 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 a peer with higher equal split 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  $|A_i|/(1 - B(u_i))$ . A 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.

We derived the matching time for BT clients with different uploading capacities (cf. Appendix A.1, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.94>). Matching time for BT clients is as large as 10 hours and it is not negligible with respect to typical download times.

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

### 3.2 Probability of Reciprocation and Expected Download Rate

The extremely long convergence time toward a stable matching has encouraged the design of subtle techniques [16] to exploit peers until a global matching is reached; then 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 [16]. 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 randomly selected peer is willing to reciprocate to peer  $i$  at the  $k$ th interaction is

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

and the expected number of peers not reciprocating peer  $i$  ( $\bar{R}_i(k)$ ) is

$$\bar{R}_i(k) = k \left( 1 - \int_0^\infty \rho(u_i, v, k) b(v) dv \right). \quad (2)$$

We simplify our analysis assuming that: 1) the number of peers not reciprocating peer  $i$  is always equal to the integer nearest to  $\bar{R}_i$  (we denote it as  $\hat{R}_i$ ) and 2) that these peers are the best uploaders of peer  $i$ . These assumptions are going to be justified by our asymptotic analysis for  $\rho(u_i, v, k)$ . In fact we are going to show that, as  $k$  diverges,  $\rho(u_i, v, k)$  converges to 0 and 1, respectively, for  $u_i < v$  and for  $u_i > v$ . Hence, at least asymptotically, peer  $i$  will not be reciprocated by all its neighbors with higher equal split (its best uploaders). This justifies assumption (2). Moreover, being that the reciprocation probability exhibits asymptotically a sharp transition from 0 to 1 at  $v = u_i$ , the number of nonreciprocating peers is distributed approximately as a binomial distribution with parameters  $k$  and  $B(u_i)$ . This justifies assumption (1), being that a binomial distribution is concentrated (in the sense of the law of large numbers) around its mean for large value of  $k$ .

Given these two assumptions, if we rank the uploaders of peer  $i$  on the basis of their equal split in decreasing order, peer  $i$  at the  $k$ th interaction will reciprocate peers with rank from  $\hat{R}_i(k) + 1$  to

$$w_i = \hat{R}_i(k) + |A_i| = \lfloor \bar{R}_i(k) + 0.5 \rfloor + |A_i|, \quad (3)$$

assuming that it is willing to open up to  $|A_i|$  connections. We use basic order statistic results [21] to derive the equal split PDF of the  $z$ th uploader of peer  $i$  (out of the  $k$  neighbors peer  $i$  has interacted with by the  $k$ th interaction):

$$b^{(z)}(v, k) = \frac{k! B(v)^{k-z} (1 - B(v))^{z-1}}{(z-1)!(k-z)!} b(v). \quad (4)$$

Note that this distribution is the same for all the peers, because optimistic unchoking selects neighbors uniformly at random.

We derive the expected download rate of peer  $j$  as:

$$\sum_{z=\hat{R}_i(k)+1}^{\hat{R}_i(k)+|A_i|} \int_0^\infty v b^{(z)}(v, k) dv + \omega \int_0^\infty v b(v) dv, \quad (5)$$

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

Finally, we derive the reciprocation probability. 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 a higher equal split than that of the  $w_j$ th uploader of peer  $j$ <sup>2</sup>, then

$$\rho(u_i, u_j, k+1) = \int_0^{u_i} b^{(w_j)}(v, k) dv. \quad (6)$$

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

1. In this work, we use the same empirical distribution as in [16].

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 (2), (3), (4), and (6) can be used to evaluate the evolution of reciprocation probabilities.

Now we are going to deepen our understanding of the system, by studying the asymptotic limit of the reciprocation probability, and the relation of these results with the matching time as defined in Section 3.1.

First, we are going to show that, as  $k$  diverges,  $\rho(u_i, v, k)$  converges to 0 and 1, respectively, for  $u_i < v$  and for  $u_i > v$ . If this is the case, then

$$\lim_{k \rightarrow \infty} \frac{\bar{R}_j(k)}{k} = \int_0^{u_j} b(v)dv = B(u_j).$$

Let us define

$$\alpha(u_j) = \lim_{k \rightarrow \infty} \left( 1 - \int_0^\infty \rho(u_j, v, k)b(v)dv \right),$$

we will then show that  $\alpha(u_j) = B(u_j)$ . We observe that  $\alpha(u)$  is a decreasing function of  $u$  (and hence in particular invertible), because the reciprocation probability  $\rho(u, v, k)$  is increasing in  $u$ . We observe that  $w_j$  behaves asymptotically as  $k\alpha(u_j)$ , hence we can apply the results for *central quantiles* (or central order statistics) [21] to  $b^{w_j}(v, k)$ , concluding that it is asymptotically distributed as a normal with mean  $B^{-1}(\alpha(u_j))$  and variance

$$\frac{\alpha(u_j)(1 - \alpha(u_j))}{k[b(B^{-1}(\alpha(u_j)))]^2} \xrightarrow[k \rightarrow \infty]{} 0.$$

Being that the variance converges to zero, we can derive from (6) that  $\rho(u_i, u_j, k)$  converges to 1 if  $B^{-1}(\alpha(u_j))$ —the mean of the Gaussian—is included in the integration range, otherwise it converges to zero. In conclusion  $\rho(u_i, u_j, k)$  converges to 0 and 1, respectively, for  $u_i < B^{-1}(\alpha(u_j))$  and for  $u_i > B^{-1}(\alpha(u_j))$ . Due to the monotonicity of  $B(\cdot)$  and  $\alpha(\cdot)$ ,  $u_i < B^{-1}(\alpha(u_j))$  if and only if  $u_j < \alpha^{-1}(B(u_i))$ . From (2) it follows that

$$\begin{aligned} \lim_{k \rightarrow \infty} \frac{\bar{R}_i(k)}{k} &= \lim_{k \rightarrow \infty} \left( 1 - \int_0^\infty \rho(u_i, v, k)b(v)dv \right), \\ &= \int_0^{\alpha^{-1}(B(u_i))} b(v)dv = B(\alpha^{-1}(B(u_i))). \end{aligned}$$

By definition this is also equal to  $\alpha(u_i)$ :

$$B(\alpha^{-1}(B(u_i))) = \alpha(u_i).$$

This equality holds for every value of  $u_i$ , then it has to be  $\alpha(\cdot) = B(\cdot)$ . We have so concluded our proof that  $\rho(u_i, u_j, k)$  converges to 0 and 1, respectively, for  $u_i < u_j$  and for  $u_i > u_j$ .

We may want to quantify the convergence time similarly to what done in the previous section for the simplified model. It is natural to consider the expected time needed for peer  $i$  to find  $|A_i|$  peers 1) with higher equal split (as for the matching time) and 2) willing to reciprocate. This second requirement makes this convergence time longer than the matching time (already several hours for high capacity clients). Hence, the refined analysis in this section points out an even longer phase during which the system can be exploited by greedy peers. At the same time, it shows that the duration of the

exploitable phase is not the only important aspect. In fact, the probability to reciprocate to peers with lower bandwidth converges to zero (the faster the lower the bandwidth is). This limits the extent to which greedy peers can exploit compliant ones. In Appendix A.2, which can be found on the Computer Society Digital Library, we quantify the possible gain by numerical simulations both of the time evolution of the probability of reciprocation and the expected download rate for different values of upload capacity.

### 3.3 Discussion

Our data agnostic analysis indicates that the exploiting BT 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. 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 BTyr. We decided to focus on BTyr because it merges several greedy techniques previously discussed in [15], [14]:

1. greedy peer set size, which implies that peer set size in BTyr is larger than that of a BT client (this approach is adopted also in BitThief [15]) resulting in a higher probability of being optimistically unchoked;
2. greedy uplink allocation, which 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 varies over time.

Here, we characterize the contributions of the BTyr building blocks to the performance achieved by a **single** BTyr client in a torrent where all the other clients are BT.

### 4.1 Simulator Description, Methodology, and Settings

Our work is based on the publicly available BitTorrent simulator called GPS [22], customized for our needs. In this environment, peers have infinite downlink capacity and a finite uplink capacity. The uplink capacity is randomly chosen according to the bandwidth distribution measured in [16].

The main performance metrics we use are:

- **Download time** of the single client (BT, BTyr, or BTold) in the different scenarios.
- **Download time** of all peers.
- **Number of pieces** uploaded by the single client during the download process.

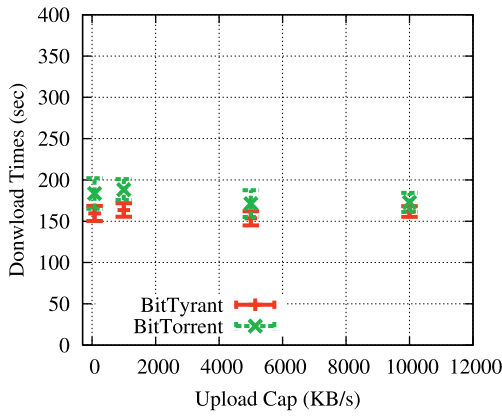


Fig. 1. Mean download time of a single client with different bandwidths (95 percent confidence interval).

We decided to follow the system parameters used in [16]: we analyze torrents of 350 peers where one initial seed shares a file of 50 MB. We chose such a file size since we noted that the gain of a strategic client is mainly *concentrated at the beginning* of the distribution process (cf. Section 3), so BTyr should benefit more from downloading smaller files.

In order to verify the results against different scenarios, we have also performed experiments with bigger file size (350 MB) and bigger torrents (500 peers) obtaining similar results.

Peers randomly start to download the content within a small interval of time (10 seconds) and stay as seeds in the system once they finish downloading the content. This scenario represents the most favorable scenario for BTyr, since peers start downloading approximately at the same time and they do not have any knowledge of the other peers' bandwidth.

We carry out a comparative performance analysis of a *single peer* using whether BT or BTyr client when the *rest of the torrent population* use BT clients. In order to make a fair comparison, at every change of the client of the single peer, we leave unchanged its characteristics (bandwidth, arrival instant), as well as the characteristics of the other peers. We estimate the mean download time over multiple runs (if not specified, we perform 10 runs for each experiment), along with the confidence interval for a confidence level of 95 percent.

#### 4.2 Impact of the Peer Set Size 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 peers. In this case, BTyr keeps contacting the tracker in order to increase the peer set as much as possible. A similar scenario has been used in the experiments showed in [16], with the difference that the torrent was composed by BTold peers. We show the results for this case (BTyr in a torrent of BTold peers) in Appendix B.1, which can be found on the Computer Society Digital Library, which are coherent to the ones found in [16].

When we consider a BTyr client in a torrent of BT peers, the performance gain of BTyr disappears. Fig. 1 illustrates the download time of a single BT and BTyr client for different classes of uplink capacity. The reason is due to the large number of active connections (active set size) established by fast peers using BT. Their uplink capacity

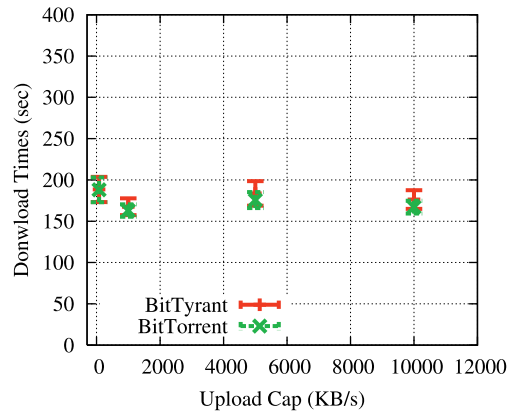


Fig. 2. Mean download time of a single client with a constrained peer set (95 percent confidence interval).

is overpartitioned, hence remote peers (including the BTyr client) receive smaller download rates as compared to the original BTold algorithm.

The BTyr itself maintains a large number of active connections. On the one hand, by keeping a larger number of active connections, BTyr strives for maximizing the chance of being reciprocated. 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. For further details, please refer to Appendix B.2, which can be found on the Computer Society Digital Library.

As additional test, we obstruct the peer set construction of BTyr: the peer set size is then equal at most to 80 for every peer in the torrent. Fig. 2 shows a slightly decreased performances of BTyr, even if we cannot exactly quantify this decrease, since the variation remains on the order of the confidence interval size. A similar result can be seen in Appendix B.1, which can be found on the Computer Society Digital Library, where we show the same experiment in the case of BTyr in a torrent of BTold peers.

These results indicates that the increased peer set size may constitute one of the factors influencing download performance.

#### 4.3 Impact of Greedy Uplink Capacity Allocation

In the previous section, we focused on the peer set and active set sizes. It remains to evaluate if the uplink capacity allocation algorithm adopted by BTyr is actually able to increase the performance. In particular, we now focus on the interaction between a peer and its peer set and ask whether peers tend to match with neighbors with similar uplink capacity in variants of the BT client.

Fig. 3 shows the ECDF of the uplink capacity of the neighbors unchoked during the whole file download by a peer with an uplink capacity of 200 KB/s.<sup>3</sup>

3. The figure also reports the uplink capacity distribution of all peers that we used in our experiments.

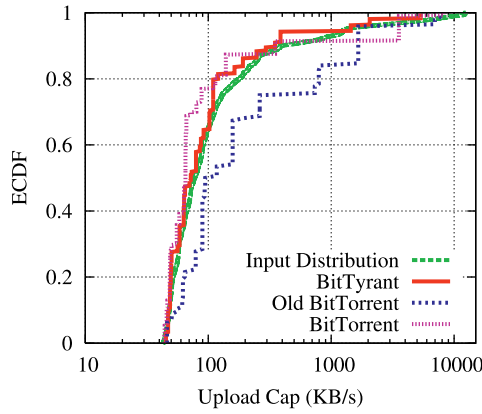


Fig. 3. ECDF of the upload bandwidths of the active set seen by a BT, BTold, and BTyr client with an uplink capacity of 200 KB/s.

We observe that a single BTold client unchokes a small subset of its neighbors (the ECDF has a few well-defined steps) and mostly reciprocates remote peers with a similar uplink capacity. Ideally, a completely stratified system would imply a single step ECDF function centered on the observed peer's uplink capacity. In practice, as discussed in Section 3, the system requires time to converge to a stable state, during which peers may end up cooperating with remote peers with different uplink capacities. In contrast to our observations on BTold, both BTyr and BT interact with any neighbor in their peer set, irrespectively of the uplink capacity, in a manner that resembles to a round robin approach. Although the intuition behind the design of BTyr was to seek and exploit for the longest possible time the fastest peers using a clever uplink allocation algorithm, no stable matching appears.

In order to understand if the dynamic uplink bandwidth allocation algorithm works properly, we take a different perspective. We neglect the effect of piece availability in our simulations: we 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.

To this aim, we show the uplink rate assigned by the BTyr peer to each neighbor, over time: for the observed BTyr peer we maintain a matrix  $E$  where the element  $e_{ij}$  represents the rate assigned to peer  $i$  at choking interval  $j$ .

Fig. 4 illustrates the matrix  $E$  for a BTyr peer  $k$  with 5,000 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. Note that since the initial rate for each unchoked neighbor is 15 KB/s, the BTyr peer unchokes *all* its neighbors.

The allocated uplink capacity varies over time, and it's possible to observe a specific trend: the bandwidth is initially equally divided among the peer set; then the peer assigns an increasing amount of bandwidth, which degenerates into a periodic, on-off, phase. A deeper analysis shows that not all the high capacity neighbors are detected or maintained. For instance, in the figure, neighbors with ID 141, 153, and 199 have high bandwidth, and they are detected, but not maintained (after 40 rounds the BTyr peer stops unchoking them). Moreover, the periodic trend, common to all the neighbors, represents an undesired behavior that limits the

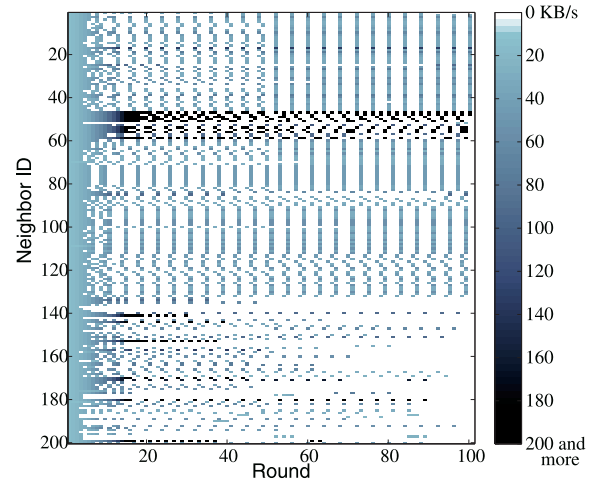


Fig. 4. Upload rate in the data agnostic case: snapshot for a fast BTyr client.

performance of the scheme: the received rate, in fact, goes periodically to zero for three rounds, forcing BTyr to try to be reciprocated again by the neighbors.

The existence of a periodic behavior hints at the presence of closed loop dynamics that the original design has not foreseen. The limited space does not allow for a deeper analysis from a control theory viewpoint, which we reserve for future works. Our extensive results illustrate an unexpected behavior of BTyr that may have an impact on the protocol performance. As a reference, the matrix representation of the upload algorithm in BTold is shown in Appendix B.3, which can be found on the Computer Society Digital Library.

## 5 THE MULTIPLE CLIENTS CASE

In this section, we study the behavior of a system in which an initial population of BT clients is gradually replaced by an increasingly larger fraction of greedy variants. This study is motivated by the fact that if the first users adopting a greedy variant experience improved performance in comparison to BT, further users can be attracted to adopt it. Since the BTyr clients have problems to interact with each other, [16] suggests to cope with this situation using a block-based Tit-For-That (TFT) strategy: however, a number of issues related to the BTyr block-based TFT choke algorithm suggest that this technique might be ineffective (cf. Appendix C.1, which can be found on the Computer Society Digital Library). Therefore, in the following we analyze the performance of multiple BTyr clients that do not implement the block-based TFT mechanism.

### 5.1 Increasing Fraction of Greedy Variants

We consider the implications of gradual user adoption of BTyr clients. We ask whether it is possible to predict an equilibrium point by which there would be no incentive for users to adopt the BTyr variant. Hence, we focus on the effects of an increasing adoption of BTyr clients in a swarm composed by BT clients on users' download times. In the following experiments, we use the same settings described in the previous sections, and introduce an increasing percentage of BTyr clients.



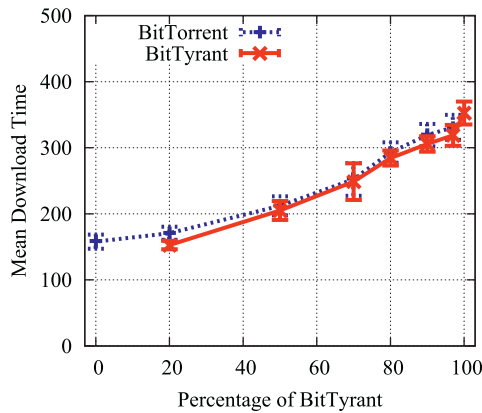


Fig. 5. Average of the mean download times for an increasing fraction of BTyr clients in a population of BT peers (95 percent confidence interval).

Fig. 5 shows this scenario. There is no performance improvement of clients switching from BT to BTyr, and no equilibrium point is found: therefore, we can conclude that BT seems to be robust against an adoption of even a small fraction of BTyr clients. Interestingly, we note that the increase of the percentage of BTyr clients has a negative impact on the mean download time of the swarm: in the next section, we dissect this issue, considering the extreme case of a pure BTyr swarm.

In Appendix C.2, which can be found on the Computer Society Digital Library, we study similar scenarios, where users with BTold clients start adopting BTyr clients, or BT.

## 5.2 The Case of Massive Adoption

We focus on the extreme scenario of massive adoption of BTyr. Fig. 6 illustrates the ECDF of the download times for a torrent of all BT 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.

To understand exactly why system performance degrades when all peers use BTyr, we analyze the upload capacity allocation of a fast BTyr peer in Appendix C.3, which can be found on the Computer Society Digital Library.

## 5.3 PlanetLab Experiments

We summarize the results for a set of experiments obtained using PlanetLab [23] as a testbed. For the details of such experiments, we refer to Appendix C.4, which can be found on the Computer Society Digital Library.

We have reproduced a similar set of experiments shown in Fig. 5 using real BT and BTyr software distributions. We note that BTyr has no gain over BT. In any case, the results confirm that BTyr clients need a favorable environment, otherwise the gain seems unpredictable.

## 6 CONCLUSIONS

Recent days have observed the development of greedy peer-to-peer clients aiming at decreasing content download times by leveraging subtle techniques to exploit generous clients. In this work, we focused on BitTorrent networks and analyzed two commonly deployed greedy techniques

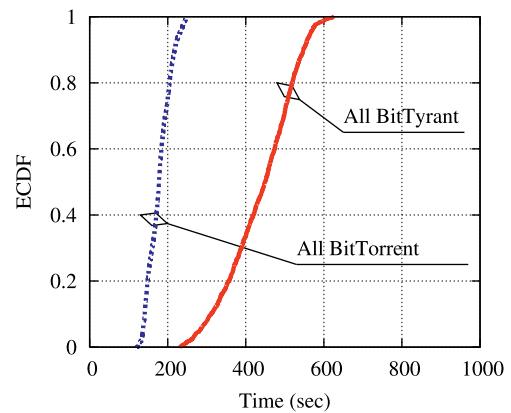


Fig. 6. ECDF of download times for BT and BTyr.

(implemented in BitTyrant). We showed that the BT protocol can be misused to gain an advantage over standard peers by building progressively larger peer sets.

We then deconstructed the other greedy building blocks—which constitutes the BitTyrant clients—and showed, on the one hand, some undesired behaviors, such as periodic chokes/unchokes, that limit the performance of the client; on the other hand the greedy uplink allocation algorithm of BitTyrant has some positive implications on the content distribution process, especially during its bootstrap phase.

Furthermore, we shifted our focus from the performance of a single BitTyrant client to the analysis of system performance when an increasing number of such greedy clients is adopted. Our results indicate that a gradual “invasion” of BitTyrant clients progressively degrades the mean download time of the swarm, suggesting that a massive adoption of BitTyrant has to be avoided. Nevertheless, we studied this extreme scenario and our results pinpoint at a severe performance degradation for all peers as a consequence of instabilities of the choke algorithm of BitTyrant, that was not designed to function in competition with other similar clients.

The instability problems of subtle variations of the choke algorithm highlighted by our work indicate that further work is required to exploit the potential benefits of alternative schemes to allocate the uplink capacity of a peer. Along the same lines, our results show that the last version of the legacy BitTorrent client is more robust than his predecessor to BitTyrant. Moreover, we noticed a significant performance improvement of BitTorrent due to a larger number of active connections for high capacity peers (cf. Appendix C.2, which can be found on the Computer Society Digital Library). Hence, the adoption of the new version of the choke algorithm of BitTorrent in all clients supporting this protocol is highly recommended.

## ACKNOWLEDGMENTS

This work has been partially funded by the ANR-VERSO VIPEER project.

## REFERENCES

- [1] B. Cohen, “Incentives Build Robustness in BitTorrent,” *Proc. First Workshop Economics of Peer-to-Peer Systems (P2P-Econ)*, June 2003.

- [2] <http://www.guardian.co.uk/technology/2006/oct/19/guardian-weeklytechnologysection.insideit>, 2011.
- [3] <http://www.theglobeandmail.com/servlet/story/RTGAM.20071128.wgtbittorrent29/BNStory>, 2011.
- [4] <http://arstechnica.com/news.ars/post/20080421-study-bittorrent-sees-big-growth-limewire-still-1-p2p-app.html>, 2011.
- [5] D. Levin, K. LaCurtis, N. Spring, and B. Bhattacharjee, "Bittorrent Is an Auction: Analyzing and Improving Bittorrent's Incentives," *Proc. ACM SIGCOMM*, Aug. 2008.
- [6] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough," *Proc. ACM SIGCOMM*, Oct. 2006.
- [7] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-Like Systems," *Proc. ACM SIGCOMM*, Oct. 2005.
- [8] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," *Proc. Passive and Active Measurements (PAM)*, Apr. 2004.
- [9] X. Yang and G. de Veciana, "Performance of Peer-to-Peer Networks: Service Capacity and Role of Resource Sharing Policies," *J. Performance Evaluation—P2P Computing Systems*, vol. 63, no. 3, pp. 175-194, Mar. 2006.
- [10] A.R. Bharambe, C. Herley, and V.N. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," *Proc. IEEE INFOCOM*, May 2006.
- [11] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM SIGCOMM*, Aug. 2005.
- [12] F. Bin, D.M. Chiu, and J.C.S. Lui, "The Delicate Tradeoffs in BitTorrent-Like File Sharing Protocol Design," *Proc. 14th IEEE Int'l Conf. Network Protocols (ICNP)*, Nov. 2006.
- [13] F. Bin, D.M. Chiu, and J.C.S. Lui, "Stochastic Differential Equation Approach to Model BitTorrent-Like P2P Systems," *Proc. IEEE Int'l Conf. Comm. (ICC)*, June 2006.
- [14] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting BitTorrent for Fun (But Not Profit)," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS)*, Feb. 2006.
- [15] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free Riding in BitTorrent Is Cheap," *Proc. Fifth Workshop Hot Topics in Networks (HotNets-V)*, Nov. 2006.
- [16] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do Incentives Build Robustness in BitTorrent?," *Proc. Symp. Networked Systems Design and Implementation (NSDI)*, Apr. 2007.
- [17] D. Carra, G. Neglia, and P. Michiardi, "On the Impact of Greedy Strategies in BitTorrent Networks: The Case of BitTyrant," *Proc. Eighth IEEE Int'l Conf. Peer-to-Peer Computing (P2P)*, Sept. 2008.
- [18] N. Laoutaris, D. Carra, and P. Michiardi, "Uplink Allocation beyond Choke/Unchoke or How to Divide and Conquer Best," *Proc. ACM Conf. Emerging Network Experiment and Technology (CoNext)*, Dec. 2008.
- [19] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and Sharing Incentives in BitTorrent Systems," *Proc. ACM Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2007.
- [20] A. Gai, F. Mathieu, F. de Montgolfier, and J. Reynier, "Stratification in P2P Networks: Application to BitTorrent," *Proc. 27th Int'l Conf. Distributed Computing Systems (ICDCS)*, July 2007.
- [21] H.A. David and H.N. Nagaraja, *Order Statistics*. John Wiley & Sons, 2003.
- [22] W. Yang and N. Abu-Ghazaleh, "GPS: A General Peer-to-Peer Simulator and Its Use for Modeling BitTorrent," *Proc. 13th IEEE Int'l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS)*, Sept. 2005.
- [23] PlanetLab, <http://www.planet-lab.org/>, Dec. 2009.



**Damiano Carra** received the Laurea degree in telecommunication engineering from the Politecnico di Milano, and the PhD degree in computer science from the University of Trento. Currently, he is working as an assistant professor in the Computer Science Department at the University of Verona. His research interests include modeling and performance evaluation of peer-to-peer networks, and distributed systems.



**Giovanni Neglia** received the master's degree in electronic engineering and the PhD degree in telecommunications from the University of Palermo, Italy, in 2001 and 2005, respectively. In 2005, he was a research scholar at the University of Massachusetts Amherst, visiting the Computer Networks Research Group. Currently, he is working as a researcher in Maestro team at INRIA. Before joining INRIA as a permanent member in September 2008, he was a post/doc

at the University of Palermo from June 2005 and an external scientific advisor in Maestro team from September 2006. His research focuses on modeling and performance evaluation of computer networks and proposals of new mechanisms to improve their performance. To this end, he applied different mathematical tools (Markov processes, control theory, fluid models, game theory, gradient-based and metaheuristic optimization). He is a member of the IEEE.



**Pietro Michiardi** received the MS degree in electrical engineering from the Politecnico di Torino and the PhD degree in computer science from the Telecom ParisTech. Currently, he is an assistant professor of computer science at EURECOM. His research interests include system, algorithmic, and performance evaluation aspects of computer networks and distributed systems. He is a member of the IEEE.



**Francesco Albanese** received the BA and MSc degrees in computer science from the Politecnico di Torino in 2006 and 2009, respectively. Currently, he is working toward the PhD degree at the Institut Eurecom, a branch of TELECOM ParisTech. His current interests include computer networks, peer-assisted technologies, and distributed databases.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).