Changing the Unchoking Policy for an Enhanced *BitTorrent*

Vaggelis Atlidakis, **Mema Roussopoulos** and **Alex Delis**
University of Athens, Athens, 15784, Greece
{v.atlidakis, mema, ad}@di.uoa.gr

***Abstract*—In this paper, we propose a modification to the *BitTorrent* protocol related to its peer unchoking policy. In particular, we apply a novel optimistic unchoking approach that improves the quality of inter-connections amongst peers and consequently, improves data distribution efficiency without penalizing underutilized and/or idle peers. Our optimistic unchoking policy takes into consideration the number of peers currently interested in downloading from a client that is to be unchoked. Our conjecture is that clients having few peers interested in downloading data from them, should be favored with optimistic unchoke intervals; this will enable the clients in question to receive data since they become unchoked faster and in turn, they will trigger the interest of additional peers. In contrast, clients with plenty of "interested" peers should enjoy a lower priority to be selected as "planned optimistic unchoked" since these clients likely have enough data to forward and have saturated their uplinks. In this context, we increase the aggregate probability that the swarm obtains a higher number of interested-in-cooperation and directly-connected peers. Therefore, we claim that an improved inter-connection of peers is achieved. Experimental results indicate that our approach significantly outperforms the existing optimistic unchoking policy.**

I. Introduction

Peer-to-peer applications remain of crucial importance as there is still a growing trend for exchange of large multimedia files [1], voice over IP [9] and broadcasting of TV-quality programs [2], [8] in the World Wide Web. Content delivery networks based on the traditional client-server model were shown not to scale for large content sharing aggregations. Most of their limitations emanate from the lack of bandwidth that causes bottlenecks in light of heavy requests. In addition, quality of service at the client side inadvertently suffers when servers experience substantial loads. In contrast, highly decentralized *peer-to-peer* models [2], [8], [1] do not distinguish the role of providers and consumers as peers play a dual role by being both a server and/or a client at times. The absence of a centralized authority also constitutes the foundation for scalable and adaptive applications.

Nowadays, *BitTorrent* is the most popular *peer-to-peer* protocol, accounting for approximately 27-55% of all Internet traffic depending on geographical location, according to [5]. In the pre-*BitTorrent* era, Napster [7], Gnutella [4] and FastTrack [3] were widely-used protocols for transferring multimedia files, such as mp3's, movies, and software. However, their centralized indexing methods and/or the lack of a *tit-for-tat* schema among peers prevented them from being an effective competitor to *BitTorrent*'s dominance.

The *BitTorrent* protocol [1] operates in three different layers: At the *swarm layer*, a peer contacts a tracker to join a swarm and receive a list of other peers to whom to connect. At the *neighborhood layer*, the core reciprocation mechanism is implemented, which forces peers to share any received data in order to receive downloading slots from counterparts. This is done locally, without any help from a centralized mechanism and constitutes the fundamental choice for the incentive policy in use. At the *data layer*, a file is viewed as a concatenation of fixed-size pieces that are requested in a rarest-first policy to ensure the highest degree of content replication. In this paper, we focus at the *neighborhood layer* and modify the neighborhood selection mechanism of the protocol known as peer unchoking; this includes regular unchoking and optimistic unchoking. Regular unchoking is the basic mechanism that implements a *tit-for-tat* schema that allocates bandwidth preferably to peers sending data and penalizes free-riders. Periodically, every peer sorts its uploaders according to the rate they provide data and sends data only to the top-three uploaders. Peers not uploading data are excluded from this process, and therefore, they receive no reciprocation. Optimistic unchoking ensures that new peers have a chance of downloading one first piece without having sent any themselves.

The question we seek to answer in this paper is how an uploader should allocate its *optimistic unchoke interval* to downloaders to achieve the most aggregate benefit in a swarm. The existing optimistic unchoking policy uses a round-robin approach giving priority to more recently connected peers [1]. This approach guarantees at least one bootstrapping interval for any new peer, regardless of the situation (i.e., dynamics) in which it finds itself. In a set of newly connected peers, some of them may already possess data blocks, while others do not. Those who possess highly-demanded data are more likely to receive data requests, thus immediately contributing to the swarm. In contrast, peers without data on high-demand or no data at all are more likely to be underutilized. Our proposal is that clients having few peers interested in downloading data, should be favored with *optimistic unchoke intervals*. In turn, this approach enables the clients in discussion to receive data since they get unchoked and so, they may trigger the interest of additional peers. To this end, we check the number of *interested* active connections a client maintains and select as the planned optimistic unchoked node the one with the least number of *interested* connections. Uploading clients with few peers interested in downloading from them, receive data in order to trigger global interest and attract block requests. In the long run, the peers in question will be rewarded with additional bandwidth from others due to regular unchoking *tit-for-tat* schema and will stop being idle. As a matter of fact, more peers will participate in the distribution of data, asserting a high quality of inter-connection of peers.

We examine a number of key factors that help our approach enhance the performance of the native *BitTorrent* protocol; they include the number of peers acting as intermediates, decongestion in seeders, contribution of aggregate seeders and peers, bootstrapping period of new peers, average downloading time and altruism presented by peers. The contributions of our work are:

1) enhancement of the *BitTorrent* protocol that collectively enables an increase in peer content contribution. A high number of peers now act as intermediaries as underutilized peers have a higher priority to receive *optimistic unchoke intervals*.
2) decongestion of seeders as fewer peers remain idle and so the load on seeders eases up considerably.
3) a shorter bootstrapping period for fresh peers compared with the native *BitTorrent* protocol.

Although related research has been carried out in a number of aspects including reciprocity mechanisms [14], [16], *tit-for-tat* schema to discourage free riding [22], and incentives policies in [20], [15], our work is to the best of our knowledge the first effort to adopt an alternative optimistic unchoking policy. Previous research has suggested solutions regarding the modification of the *regular unchoking policy*, and has introduced techniques to encourage peers to act as uploaders and to discard idle peers. Our work, however, is the very first to modify the *optimistic unchoking policy* to encourage cooperation of peers. Our purpose is to treat underutilized uploaders as nodes that lack data to upload, rather than consider them to be selfish free-riders. It is the first time that uploaders are able to locate idle peers and "reward" them with optimistic unchoking slots; no central authority point is used to locate idle peers. Our new optimistic unchoking policy increases the number of interested-in-cooperation and directly-connected peers. In this manner, the quality of inter-connection of peers is improved and a high number of peers now act as data intermediaries, rather than remain idle. Via experimental evaluation and comparison of our protocol with the native *BitTorrent*, we show a significant increase in uploading bandwidth offered by peers and a decrease in the amount of time peers remain idle. We also show that a noteworthy number of peers upload more blocks than download, so we claim that our protocol modification yields an increase in altruism presented by peers.

The rest of the paper is organized as follows. Section II briefly presents prior related work and Section III provides necessary background on the *BitTorrent* protocol. Section IV outlines the key features of our proposed optimistic unchoking scheme. Section V presents our main experimental results and Section VI offers our concluding remarks.

II. Related Work

A number of techniques have been suggested to improve the performance of the native *BitTorrent* [12] protocol including bartering-based approaches among peers, and incentive-based policies. In [14], [16], indirect and direct reciprocity mechanisms are examined so that peers exploit their own data contributions to obtain data from others. Our approach differs from the above efforts as we suggest an unchoking policy in which peers do not exploit their contribution to obtain data. Under our enhanced *BitTorrent* protocol, peers altruistically offer data to underutilized and/or idle counterparts. In [20], the issue of incentive compatibility was re-examined. The authors showed that even though the tit-for-tat approach was intended to discourage free-riding, the performance of *BitTorrent* has very little to do with this fact. Also through the release of the *BitTyrant* client, the conjecture of whether incentives build robustness in *BitTorrent* is evaluated. Incentives in *BitTorrent* systems are also studied in [15], where the unchoking algorithm of native *BitTorrent* is evaluated. This work shows that regular unchoking facilitates the formation of clusters of peers with similar bandwidth, which is also the case in our enhanced *BitTorrent* protocol.

A variety of mechanisms for preventing free-riding in *P2P* file-sharing systems are applied in [22], [25], [19]. Although applying mechanisms to discourage free-riding is essential to steering more peers to act as data intermediaries, it is not the answer to reducing the bootstrapping period of peers with no initial data to upload. With our improved unchoking policy, uploaders immediately locate and furnish data to peers with no initial data blocks.

Neighborship consistency is defined as the ratio between the number of known nodes and the number of actual nodes within a node's area of interest and can be used to measure the quality or connectivity in *P2P* systems [13]. In response to neighborship, in our enhanced *BitTorrent* protocol we increase the aggregate ratio of interest of peers (Section IV-B). In [23], the use of altruism in *P2P* networks is examined; altruism is defined via a parameter that reflects benefit obtained for a peer's contribution. A peer selectively decides the level of its own contribution and demands to download a specific amount of data; the amount of data a peer demands is proportional to its contribution. In our enhanced *BitTorrent* approach, a peer decides which peer to unchoke in order to maximize its ratio of interest. Benefit obtained from our unchoking policy is examined collectively. We increase the number of directly-connected and interested-in-cooperation peers in an attempt to build a robust swarm.

Tracker-related bootstrap problems are addressed in [24]. A tracker may present a bottleneck in light of many peers joining a swarm simultaneously. A solution to this problem is to register *.torrent* metadata files with many trackers. However, [24] does not examine the bootstrap issue for peers that initially hold no data to upload. In contrast, our approach achieves the shortest possible bootstrapping period for peers without initial data.

There have also been proposals for new models that essentially suggest *BitTorrent*-like protocols [11], [21], [18], [19], [17]. However, our work is the first to suggest a modification to the optimistic unchoking policy that collectively increases the number of peers acting as intermediaries, decongests seeders, and decreases the bootstrapping period of peers.

III. Background

This section provides the necessary background for an in-depth understanding of the *BitTorrent* protocol.

A. Terminology

Files transfered using the protocol are split into identical-sized *pieces*, typically between 32 KB and 4 MB with each such *piece* further split into *blocks*. The protocol deals with the distribution of *pieces* whose *blocks* are ultimately transported with *TCP*. A *peer* is an instance of a *BitTorrent* client that runs locally on a machine. Usually, *leecher* is the term given to a client not possessing a whole copy of the file; *seeders* are peers that maintain an entire copy of the file and provide content without doing any downloading. *Leechers* or *leeches* are essentially downloaders while *seeders* are uploaders. All peers sharing a file including *seeders* and *leechers* make up a *swarm*.

The *peer unchoking policy* is a tit-for-tat schema adopted by peers to ensure a proper downloading and uploading reciprocation. The term *interested* describes a peer wishing to receive data from another peer and *choked* describes a peer to whom an uploader refuses to send data. The *tracker* is the only central point of authority and its main role is to provide peers with contact information of others to whom to connect. The *initial seeder* is the client that creates a *.torrent* metadata file and publishes it on selected websites; the latter act as global directories of available files [6].

B. Tracker

A tracker for a specific file keeps a global registry of all the downloaders and seeders and enables peers to locate each other and commence downloading. The tracker does not provide access to any downloadable data itself. It simply maintains a who-is-who currently involved in the distribution of each file and collects statistics on the dynamics (i.e., activity) of the swarm. Periodically, the tracker sends to every peer an updated list of swarm participants. As soon as initial contact information is received, a peer can continue downloading without the intervention of a tracker. A tracker may also collect statistics on the distribution of different files and coordinate multiple swarms at times. Peers communicate with the tracker via plain text messages using *HTTP* and port *6969*. Contact addresses are registered in the *.torrent* metadata file. Alternatively in systems without a tracker, every peer also acts as a tracker; this is the case with *Azureus* [10] that uses *DHT*s.

C. Peer Selection – Unchoking Algorithm

The *unchoking* algorithm implements a *tit-for-tat* schema where peers preferentially dispatch data to peers that send data back. The algorithm is invoked every 10 seconds as well as every time a peer disconnects from the local client and when an unchoked peer becomes *interested* or *uninterested*. The local host sorts peers having sent data over according to their uploading rate every time the algorithm is invoked and keeps unchoked the top-three uploaders called *regular unchoked*. Also at the beginning of every three rounds, an additional peer is selected at random, called *planned optimistic unchoked*, to be kept unchoked for 30 seconds, regardless of its uploading contribution. *Optimistic unchoking* guarantees that a new peer will be able to download at least one piece, without having sent any.

A time interval of 10 seconds is used to avoid *oscillation*, a situation where connections are choked and unchoked so quickly that clients are unable to exchange data; 10 seconds is a long enough bootstrapping period for a *TCP* connection. Note that *more* than 4 peers may be unchoked at a time but *only* 4 of those are interested. It is worth pointing out that the algorithm is called every time an unchoked peer gets interested or uninterested, sorts uploaders according to pieces received and keeps unchoked the three fastest of them and the planned optimistic unchoked. If the planned optimistic unchoked is part of the above set of three peers (*regular unchoked*), a new peer is chosen and unchoked repeatedly until an interested peer is identified.

D. Peer States

Each peer must maintain two state flags for each end of the connection, namely, choked and interested. Since connections are bidirectional, four such flags are maintained by each peer:

- **choking**: The client as an uploader chokes the remote peer (downloader). The client must discard any unanswered requests of the remote peer.
- **having-an-interest**: The client as a downloader is interested in receiving pieces from the remote peer (uploader).
- **choked**: The client as a downloader is choked by the remote peer (uploader). A client is not expected to send any data-oriented messages to a choked connection, but it sends state-oriented messages.
- **interested**: The remote peer (downloader) is interested in receiving pieces from this client (uploader).

E. Piece Selection

The piece selection strategy consists of three facets: *Random-First-Piece*, *Rarest-First* and *End-Game Mode*. At the beginning, a peer has nothing to upload so the first four pieces are selected at random to commence downloading and then the rarest-first-policy is applied. Local peers maintain statistics about the number of copies each data-piece has in the swarm so the rarest-first-policy can be applied. A piece may be marked as rarest any time a copy of another piece is added to or removed from the data set of the local peer. The *Rarest-First* discipline effectively increases the degree of content replication among peers. Finally, the *End-Game Mode* is used at the very end of downloading when a peer has requested all pieces. During this mode, a peer requests all blocks of partially downloaded pieces from all members in its peer-set; the peer also cancels its requests upon receiving the pieces in question. This is done because completion of downloading should not be delayed by a single missing block to be received through a low-capacity or idle connection.

F. Overview of BitTorrent *Operation*

At first, a peer downloads a *.torrent* metadata file to contact a tracker and receive a list of 50 peers with whom to establish *TCP* connections. These *.torrent* metadata files can be found at a variety of sites such as Mininova [6]. After joining a swarm, a peer with no data to upload is interested in downloading data from every participant in this swarm. The peer will receive its first data piece during its first optimistic unchoke interval and then it will have data to upload and cooperate with others. During the execution of a client, the set of peers that the client receives data from is not necessarily identical to the group of peers[1] that the client sends data to. Data requests are serviced in a rarest-first policy and neighborhood adaption is carried out by the unchoking policy deployed. Depending on the data uploaded, a peer will receive data as a reciprocation from others or remain choked. A peer maintains no more that 40 initiated connections and uploads to a maximum of 4 of them. If the number of neighbors a peer has falls bellow 30, the peer requests a new list of peers to whom to connect. Upon receiving a full copy of the file, the peer may be selfish and depart the swarm or be an altruist and become an additional seeder. Data integrity is preserved using SHA-1 hashes, as the *initial seeder* creates a hash for every data piece and registers it in its *.torrent* metadata file.

IV. Enhanced *BitTorrent*

In this section, we outline our proposed peer unchoking policy by first introducing the messages used by our enhanced *BitTorrent* protocol. We then discuss the metrics used for maximizing the ratio of interest and the algorithms used for our unchoking policy. We give a mathematical model that corresponds to *BitTorrent*-like content sharing protocols, and we discuss the rationale for our approach to achieve shorter bootstrap time. Finally, we give an overview of our enhanced *BitTorrent* system.

A. Enhanced BitTorrent *Messages*

To implement our enhanced *BitTorrent* protocol we use three different types of messages, namely: swarm-oriented, state-oriented and data-oriented messages. Table I summarizes the *Swarm-oriented* messages that are exchanged between peers and the tracker. These messages are helpful to the tracker so that it can maintain an up-to-date mapping of the dynamics of the swarm. *Swarm-oriented* messages are also helpful to peers to help them locate each other in a timely fashion. The messages in this group contain no downloadable data.

The group of messages sent among cooperating peers is depicted in Table II. We refer to these as *state-oriented* messages that help achieve cooperation among peers and implement the peer unchoking policy. All messages of Table II contain no downloadable data but designate when peers must exchange data or not (Section III-D). More specifically when peer A dispatches a **choke** message to peer B, the latter must not send any *data-oriented* messages back to A. B must receive an **unchoke** message from A in order to commence sending new *data-oriented* messages. Furthermore, a peer will send an **unchoke** message only to remote peers that have previously sent an **interested** message. Peer A is *interested* in receiving data from peer B, if B possesses data pieces that A does not possess. **Have** and **bitfield** messages indicate the arrival of a new piece and the set of pieces possessed by a peer, respectively.

join: A peer interested in joining a swarm sends this message to the tracker. This message contains metadata of the respective file and contact information of the sender
join_response: The tracker sends this message in response to **join**; no payload.
peerset: A peer sends this message to the tracker to request the contact information of other peers participating in the swarm; no payload.
peerset_response: The tracker sends this message in response to **peerset**. This message contains a list of listening IP–addresses and ports of peers participating in the swarm.
leave: A peer sends this message to inform the tracker that it is leaving the swarm.

TABLE I
SWARM-ORIENTED MESSAGES

choke: Peer A sends this message to remote peer B to inform B that it is choked by A. Consequently, B must not send any *data-oriented* messages to A; no payload.
unchoke: Peer A sends this message to remote peer B to inform B that it is no longer choked by A. Consequently, B may send *data-oriented* messages to A; no payload.
interested: Peer A sends this message to remote peer B when A is interested in receiving data from B; no payload.
have: Peer A sends this message to every connected remote peer to inform it that it has received a new piece or to acknowledge the sender of a piece. The payload of this message is two integers indicating received piece and number of interested connections in A.
bitfield: Peer A sends this message after establishing a new connection to inform remote peer B about pieces it possesses; variable length payload that is a bitmap indicating valid blocks of A.
handshake: Peer A sends this message to establish connection with peer B. Payload includes file identifier and peer identifier of peer A.

TABLE II
STATE-ORIENTED MESSAGES

Finally, Table III summarizes the *data-oriented* messages that are sent between *unchoked* peers (i.e., peers that are exchanging data).

B. Peer Unchoking - Maximizing Ratio of Interest

We define the ratio of interest RI_p of a peer p to be $RI_p = \frac{int_p}{act_p}$, where int_p is the number of interested connections p maintains and act_p is the number of active connections of p. The number of interested connections maintained by a peer may help project the number of data requests the peer in question will ultimately receive provided that data requests are received only via active connections marked

[1]three regular unchoked and one optimistic unchoked

request: The sender of this message includes 3 integers denoting requested piece, block within piece and block length.
piece: The sender of this message includes an integer that is the position of requested piece, block's offset within piece and requested data block.
cancel: The sender of this message informs the recipient that it is no longer interested in a previously requested block of a piece. Payload consisting of 3 integers indicating piece index, block offset and block length.

TABLE III
DATA-ORIENTED MESSAGES

as interested. It is evident that peers with a low ratio of interest receive few data requests and it is likely that they are underutilized and/or idle. The unchoking policy of our enhanced *BitTorrent* protocol maximizes the aggregate ratio of interest of peers. Every time an *optimistic unchoke* is to be performed, we select the peer p with the minimum RI_p to be the *planned optimistic unchoked* peer. In the long run, our optimistic unchoking policy is effective as idle peers initially unable to act as intermediaries and content replicators will be unchoked earlier than in the native *BitTorrent* protocol where the unchoking policy is based on random choice. The peers that have saturated their uplinks will be decongested as more clients will act as content intermediaries. We anticipate that our approach will be most effective when we rotate the *planned optimistic unchoked* peer in a a prioritized way, yielding the right-of-way to fresh peers and peers with minimum interest ratios. To the best of our knowledge this is the first time such a technique is suggested. Our suggested approach does not bypass the *tit-for-tat* schema, since it does not modify regular unchoking; it rather offers an alternative to improve the quality of inter-connection of peers. An improvement in the quality of inter-connection is attained as soon as an increase in the number of directly-connected and interested-in-cooperation peers is achieved. The benefit obtained from our approach is demonstrated in section V where we compare the unchoking policies of our enhanced *BitTorrent* and the native *BitTorrent* protocol.

C. Algorithms

To apply our peer unchoking policy, we augment the *have* state-oriented message in the native *BitTorrent* with an additional float value. The latter corresponds to the ratio of interest RI of the sender of the *have* message. Our enhanced *BitTorrent* protocol invokes Algorithms 1 and 2 when a client is in *leech* and *seed* state respectively. These two algorithms are invoked every 10 seconds, every time a peer disconnects from the local client, and when an unchoked peer becomes interested or uninterested. The above timing and event-driven settings are inline with the directives of the *BitTorrent* protocol [1]. As soon as these two algorithms are invoked, a "new round" starts; the number that designates a round ranges from 1 to 3.

Algorithm 1, invoked when a peer is in leech state, takes as input the set of remote peers D currently downloading from

Algorithm 1 peer unchoking algorithm for client in *leech* state

```
Input: U, D, $RI_{p \in D}$
 1: if $round = 1$ then
 2:    sort D according to $RI$ { optimistic unchoking }
 3:    set OU
 4:    unchoke OU
 5: end if
 6: sort U { regular unchoking }
 7: set RU
 8: for $p \in D$ do
 9:    if $p \in RU$ then
10:       unchoke p
11:    else
12:       choke p
13:    end if
14: end for
15: if $OU \subseteq RU$ then
16:    repeat
17:       choose $p \in D$
18:       unchoke $p$
19:    until $p$ is interested
20: end if
```

the local client, the set of remote peers U currently uploading to the local client and the vector RI_p denoting the ratio of interest of each remote peer p. No explicit output is produced. The effect however of the algorithm is the realization of our suggested *peer unchoking policy*. RI_p vector is updated every time a *have* message is sent from a remote peer p to the local client. Peers having sent data to the local client are sorted according to their uploading rate and the top three are kept unchoked, called *regular unchoked peers* (RU). Every third round, the remote peer with minimum RI is selected as *planned optimistic unchoked* (OU) and kept unchoked from the local client (for 30 seconds). If *planned optimistic unchoked* is a member of the regular unchoked peers, a new interested peer must be added to the regular unchoked set. Note that uninterested peers may be selected unchoked until an *interested* peer is added to the regular unchoked set. However, only four *interested* peers remain unchoked in the same round.

Algorithm 2, invoked when a peer is in seed state, takes as input the set of remote peers D currently downloading from the local client as well as the vector RI_p. Again no explicit output is returned. In this algorithm, U is an empty set since the local client is in seed state and no peer is uploading to it. Peers with pending block requests are sorted according to the time they were last unchoked (most-recently-first). Remaining peers are sorted according to their downloading rates (those displaying highest rates are given priority), and are appended to the above set of sorted peers. During two rounds (out of three), the algorithm keeps unchoked the three first peers (RU); moreover, it keeps unchoked the peer p with the minimum RI_p (OU). In the third round, the algorithm keeps unchoked the first four peers (RU).

Algorithm 2 peer unchoking algorithm for client in *seed* state

```
Input: D, $RI_{p \in D}$
 1: sort D according to last unchoke time
 2: sort rest peers according to downloading rate
 3: if $round = 1, 2$ then
 4:     set RU {3 regular unchoked peers}
 5:     sort D according to $RI$
 6:     set OU
 7:     unchoke OU
 8: else
 9:     set RU {4 regular unchoked peers}
10: end if
11: for $p \in D$ do
12:     if $p \in RU$ then
13:         unchoke p
14:     else
15:         choke p
16:     end if
17: end for
```

N	Number of peers in swarm.
U	Maximum uploading rate.
D	Maximum downloading rate.
RI_i	Ratio of interest of class-i leechers.
l	Number of leechers in swarm.
s	Number of seeders in swarm.
h	Number of node classes.
l_i	Number of class-i leechers.
x_{reg}	Number of simultaneous regular unchokes each node performs.
x_{opt}	Number of simultaneous optimistic unchokes each node performs.
d_i^{reg}	Downloading rate of a node due to regular unchokes.
d_i^{opt}	Downloading rate of a node due to optimistic unchokes.
d_i^{seed}	Downloading rate of a node due to seeders' uploading.

TABLE IV
MODELING PARAMETERS

D. A Mathematical Perspective

In what follows we present a mathematical model that incorporates basic system parameters of *BitTorrent*-like content sharing protocols. We give equations that encapsulate delicate performance characteristics of the native and the enhanced *BitTorrent* protocol. Our purpose is to help the reader obtain an analytical perspective on how our approach differs from the native *BitTorrent* protocol. Parameters used in the following analysis are explained in Table IV. For clarity of presentation we make the following assumptions:

- All nodes have the same maximum uploading rate U and the same maximum downloading rate D.
- Instant uploading/downloading rate of a node is protocol related and depends on the class (defined below) to which the peer belongs.
- We assume perfect clustering between nodes, i.e., that during regular unchokes, data flows only between peers of the same class (tit-for-tat).

There are N cooperating peers, where l peers are leechers and s peers are seeders:

$$N = l + s \tag{1}$$

At global scope, the aggregate downloading rate of leechers must be equal to the aggregate uploading rate of leechers and seeders:

$$\sum_{l \in S} d_l = \sum_{l \in S} u_l + \sum_{s \in S} u_s \tag{2}$$

Let h be the number of node classes, where each class is defined by RI_i and there are l_i class-i leechers. Ratio of interest RI_i indicates the number of utilized active connections of class-i leechers and designates the instant uploading rate u_i of the leechers in question. Then, for class-i leechers it is:

$$u_i = U \cdot RI_i \tag{3}$$

and

$$N = \sum_{i=1}^{h} l_i + s \tag{4}$$

If we take into account clustering of leechers in classes, equation (2) becomes as follows:

$$\sum_{i=1}^{h} u_i \cdot l_i + \sum_{s \in S} u_s = \sum_{i=1}^{h} d_i \cdot l_i \Rightarrow$$

$$\sum_{i=1}^{h} RI_i \cdot U \cdot l_i + \sum_{s \in S} u_s = \sum_{i=1}^{h} RI_i \cdot D \cdot l_i \tag{5}$$

Regular unchokes

We assume perfect clustering, i.e., that during regular unchokes data flows only between peers of the same class (tit-for-tat), and consequently from (5) we get: $RI_i \cdot U \cdot l_i = D \cdot l_i \Rightarrow d_i^{reg} = RI_i \cdot U \cdot \frac{x_{reg}}{x}$, where x_{reg}/x is the fraction of regular unchokes out of total simultaneous unchokes a node performs. We use (3) to get the downloading rate of class-i leechers due to regular unchokes:

$$d_i^{reg} = u_i \cdot \frac{x_{reg}}{x} \tag{6}$$

Equation (6) yields that d_i^{reg} of a leecher is independent of the total number N, of peers in the swarm. This is to be expected, as the reciprocation a peer receives is only a function of the peer's uploading contribution.

Optimistic unchokes

Under the native *BitTorrent* protocol optimistic unchokes are uniformly distributed among leechers of all classes. In the same spirit, we claim that the downloading rate of class-i peers due to optimistic unchokes is:

$$d_i^{opt} = \frac{N \cdot U \cdot \frac{x_{opt}}{x}}{l} \tag{7a}$$

Under our enhanced *BitTorrent* protocol, optimistic unchokes are not uniformly distributed among leechers of all classes. Class-1 leechers who have the lowest ratio of interest are expected to receive the highest number of optimistic unchokes. In contrast, class-N leechers, who have the highest ratio of interest, are expected to receive the lowest number of optimistic unchokes. We define $c_i = f(RI_i, l_i)$ to be the fraction of optimistic unchokes allocated to class-i leechers, with $\sum_{i=1}^{h} c_i = 1$ and $c_1 > c_2 > c_3 > ... > c_h$. Then, we claim that:

$$d_i^{opt} = c_i \cdot \frac{N \cdot U \cdot \frac{x_{opt}}{x}}{l} \tag{7b}$$

Seed unchokes

We assume that the aggregate uploading rate of seeders U_s is uniformly allocated to all leechers; this assumption is inline with the latest *BitTorrent* protocol [1] and with our enhanced *BitTorrent* protocol. It is: $U_s = s \cdot U \Rightarrow d_i^{seed} = \frac{U_s}{l}$, and thus, the downloading rate of class-i peers due to the seeders' uploading contribution is:

$$d_i^{seed} = U \cdot \frac{s}{l} \tag{8}$$

Downloading rate

The total downloading rate of a class-i node is the sum of downloading rates presented above: $d_i = d_i^{reg} + d_i^{opt} + d_i^{seed}$. From (6), (7a), (7b), (8) we get the total downloading rates for native and enhanced *BitTorrent* respectively:

$$d_i^{BT} = U \cdot RI_i \cdot \frac{x_{reg}}{x} + \frac{N \cdot U \cdot \frac{x_{opt}}{x}}{l_i} + U \cdot \frac{s}{l} \tag{9a}$$

$$d_i^{EN\_BT} = U \cdot RI_i \cdot \frac{x_{reg}}{x} + c_i \cdot \frac{N \cdot U \cdot \frac{x_{opt}}{x}}{l} + U \cdot \frac{s}{l} \tag{9b}$$

Equation (9a) differs from equation (9b) only in the term that is related to optimistic unchokes. Specifically, under the native *BitTorrent* protocol, optimistic unchokes are distributed among leechers of all classes in an unbiased manner. In contrast, under the enhanced *BitTorrent* protocol a different fraction of optimistic unchokes c_i are allocated to class-i leechers. The peer unchoking policy of the enhanced *BitTorrent* protocol dynamically adapts c_i, so that the proper proportion of optimistic unchokes is allocated to class-i leechers, and prevents them from remaining idle.

E. Bootstrapping Period

In this subsection we outline optimistic unchoking and explain when our approach achieves a shorter bootstrapping period than the native *BitTorrent*. In Figure 1 we illustrate a swarm with three different sets of peers: seeders, initial leechers and random peers. The set of random peers are further split into set A and set B, according to the particular dynamics of the peers. Specifically, set A consists of peers with no data at all and set B consists of peers with some data. As presented in Figure 1, seeders join the swarm until $t0$, initial leechers join the swarm until $t1$ and random peers (set A and B) join the swarm after $t1$. Each seeder allocates its first optimistic unchoke interval to one leecher of the set of initial leechers. We assume that the initial number of leechers is equal to the number of seeders. In the native *BitTorrent* protocol and in our enhanced *BitTorrent* protocol, any peer $p \in A \cup B$ that joins swarm at $t > t_1$ must wait $W_p \leq 30sec$ to receive an optimistic unchoke interval from an uploader. In the native *BitTorrent* protocol, there is no distinction between peers of sets A and B, since planned optimistic unchoked is randomly rotated among new peers [1]. In our enhanced *BitTorrent* protocol, the unchoking policy is implemented with Algorithms 1 and 2 which are based on the RI of peers. Therefore, peers of set A will be unchoked prior to those of set B. The latter, who already possess data, will receive reciprocation after uploading, and thus will stop being idle. On the contrary, peers of set A will idle until they receive an optimistic unchoke interval $(\sum_{p \in A} W_p \gg \sum_{p \in B} W_p)$. Our schema prevents peers of set A from experiencing a lengthy bootstrapping period and does not degrade the performance of peers of set B. As a result, there is a shorter bootstrapping period in comparison with the native version of *BitTorrent*.

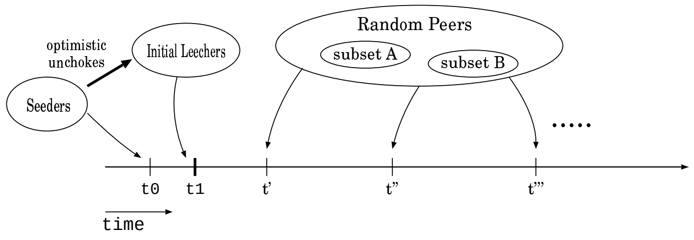


Fig. 1. Swarm dynamics as time progresses. Seeders join the swarm until t_0 and allocate optimistic unchoke intervals to initial leechers. Random peers (subsets A and B) join swarm after initial leechers.

F. Overview of enhanced BitTorrent

The *initial seeder* publishes to the *tracker* the *.torrent* file including metadata describing the file to be distributed. The *initial seeder* possesses a full copy of the designated file and is the first uploader in the swarm. A *fresh* peer wishing to join the swarm must contact the tracker (HTTP plain text messages) to obtain the .torrent file and a list of peers, referred to as *peer set*, to whom to connect. Afterwards, the fresh peer establishes TCP connections with all peers in its peer set. Each peer is multi-threaded and asynchronously downloads/uploads data from/to multiple counterparts. Enhanced *BitTorrent* peers maintain bitmaps to keep track of missing and obtained data pieces; pieces are requested using the rarest-first policy. Uploaders maintain a vector of *ratios of interest* of all peers. *Optimistic unchoking* is a process that "rewards" *underutilized* and/or *idle* peers with *optimistic unchoke* slots. Fresh peers are also rewarded with optimistic unchoke intervals from our unchoking policy to acquire initial data. Our purpose is to prevent peers with low ratios of interest from being idle and to motivate them to act as *data intermediaries*. Furthermore, the *regular unchoking* policy facilitates the formation of clusters of peers with similar bandwidth. Upon completion of downloading, each peer reports its downloading statistics for the file to the tracker, and may be selfish and leave the swarm or altruistic and become an *additional* seeder.

V. Evaluation

To evaluate our enhanced *BitTorrent* protocol, we have implemented in *Python* a respective client as well as a tracker. Our implementation of both the client and the tracker run in *Windows7, Linux* and *MacOS*. For our experiments, we used 40 workstations, each featuring a 1*GHz* clock and 1*GB* memory running *GNU/Linux*. The workstations are attached to a local Ethernet network running at 100*Mps*. Our key experimental objectives were to:

1) measure the number of directly-connected and interested-in-cooperation peers to compare the *quality* of peer inter-connections for both our enhanced and the native *BitTorrent*.
2) measure the decrease in the bootstrapping period achieved by leechers in enhanced *BitTorrent*.
3) examine pieces uploaded from leechers and seeders to evaluate the decongestion of seeders achieved by our enhanced *BitTorrent*.
4) ascertain the degree of altruism attained by leechers in our enhanced *BitTorrent*.

We used a number of key parameters that we outline in Table V. We also experimented with numerous settings regarding the distribution of seeders and leechers among our clients. In all our experiments, seeders joined swarms before leechers; the former had a full copy of the file to be distributed, while the latter had no data at all.

Optimistic unchoke interval	30 seconds
Regular unchoke interval	10 seconds
Regular unchoked peers	3
Optimistic unchoked peers	1
Active connections	40
Swarm size	$\approx$ 150 peers
Data pieces in file	1280
Piece size	$512KB$
File size	$\approx 700MB$

TABLE V
Experimental Parameters

Ratio of Interest

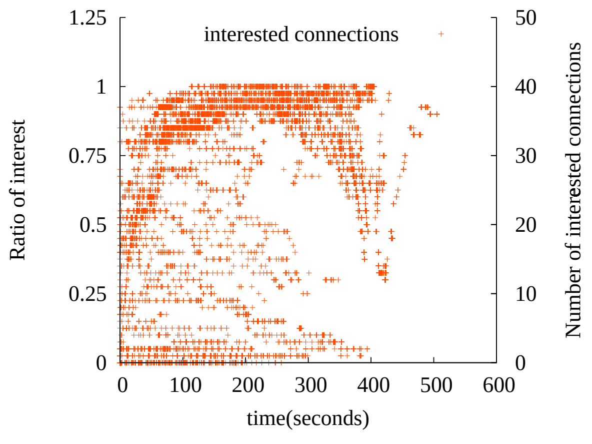
In this section, we examine the *ratio of interest* of peers, as defined in section IV-B. From a peer's local perspective, the ratio of interest indicates the amount of data requests a peer will receive from others. From a global perspective, the ratio of interest reflects the *quality* of inter-connection of peers. In this regard, the benefit of our approach is depicted by Figures 2(a) and 2(b) that illustrate the ratios of interest and number of *Interested* connections maintained by peers over the duration of the experiment. In both cases, swarms are formed from 130 leechers and 15 seeders; 90% of peers join a swarm within 100 seconds. In Figure 2(a), which corresponds to the enhanced *BitTorrent* protocol, the average ratio of interest is 0.30 per peer, while in Figure 2(b), which corresponds to the native *BitTorrent* protocol, the average ratio of interest is 0.22 per peer. Moreover, before 500 seconds in Figure 2(a) (enhanced *BitTorrent*), there is a higher coverage of *interested* connections than that of Figure 2(b) (native *BitTorrent*). In the first case, all peers act as intermediaries (downloading **and** uploading) and the ratio of interest is high until the completion of downloading. After completion of downloading, the ratio of interest is uniformly decreased. In the second case, there are underutilized peers with a low ratio of interest. This ratio of interest of idle peers becomes even lower and asymptotically reaches zero as soon as the majority of peers completes downloading. As a matter of fact, the enhanced *BitTorrent* displays a higher number of directly-connected and interested-in-cooperation peers than its original counterpart. An improved inter-connection of peers is achieved as the new unchoking policy, as implemented by Algorithms 1 and 2, maximizes the ratio of interest and provides idle peers with data. In turn, idle peers act as additional data intermediaries and "trigger" the interest of other peers. In contrast, the unchoking policy of the native *BitTorrent* protocol has no mechanism to locate idle peers and essentially does not "prod" them to cooperate with others.

Exchanging Interested-Messages

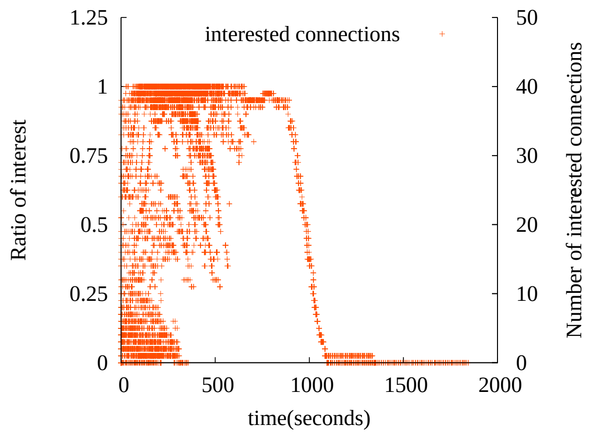
In this section, we examine the density of *interested* messages exchanged within a swarm to evaluate the quality of peer inter-connection. Peer A sends an *interested* message to peer B, when A is interested in receiving data from B (Section IV-A). A peer that is receiving many *interested* messages will ultimately contribute uploading capacity to the swarm as a whole and so we expect that it will not remain idle. In contrast, a peer that is receiving few *interested* messages will in all likelihood not act as an uploader and shall remain idle. Figures 3(a) and 3(b) show the number of interested messages received by leechers as a function of time. We show statistics collected from our enhanced *BitTorrent* leechers in Figure 3(a) and from native *BitTorrent* leechers in Figure 3(b). In Figure 3(a), an average of 200 *interested* messages is received from each leech, while in Figure 3(b), this average stands at 40 messages only. Under our proposal, leechers exchange five times more *interested* messages than under the *BitTorrent* protocol. This indicates an increase, by a factor of five, in the number of directly-connected and interested-in-cooperation leechers. Moreover, Figures 4(a) and 4(b) show that messages received by seeders under our protocol and under the native *BitTorrent* protocol follow the same pattern. In both cases there is an increase in the number of exchanged messages during the first 100 seconds as peers rapidly join the swarm. Throughout the experiment, each seeder receives about 70 *interested* messages. This is strong indication that the enhanced *BitTorrent* helps improve the quality of inter-connections amongst leechers without affecting the quality of inter-connections between leechers and seeders.

Idle vs. Active time

In this section, we examine the time that leechers (a) remain idle and (b) act as data intermediaries that facilitate uploads

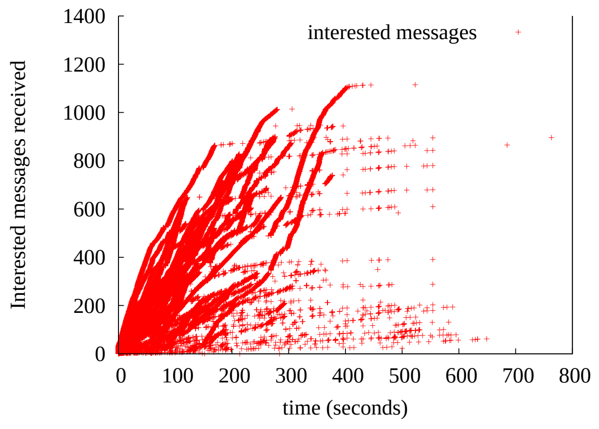


(a) Enhanced *BitTorrent*

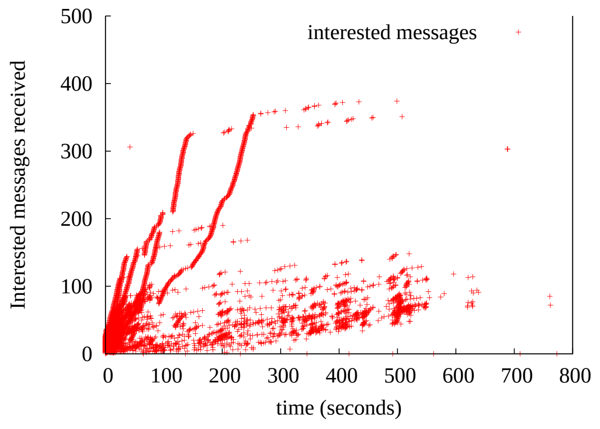


(b) native *BitTorrent*

Fig. 2. Ratios of interest of peers and *Interested* connections under (a) our enhanced and (b) the native *BitTorrent* protocol. Maximum active connections maintained per-peer are fixed at 40 and the ratio of interest is $RI \leq 1$ for both cases. The average ratio of interest is at 0.30 and 0.22 in (a) and (b) respectively.



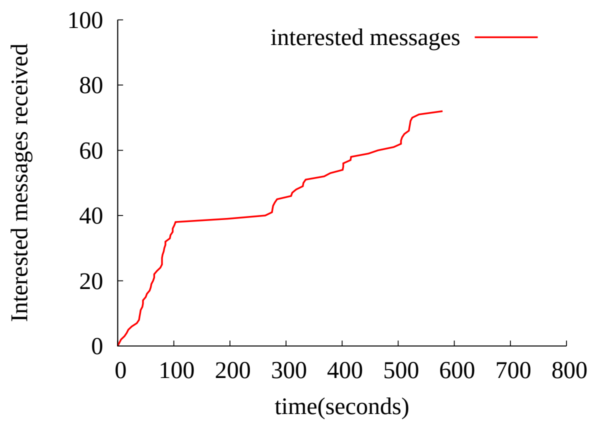
(a) enhanced *BitTorrent*



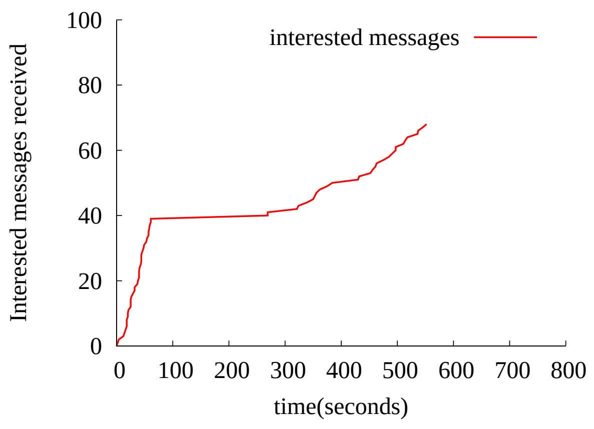
(b) native *BitTorrent*

Fig. 3. *Interested* messages received by leechers under (a) the enhanced and (b) the native *BitTorrent* protocol. In (a), leechers receive more than 1200 *interested* messages, while in (b), leechers receive at most 300 *interested* messages. The average values of 200 and 40 messages for each leecher are observed in (a) and (b) respectively.

and/or downloads. To this end, we create two swarms formed under flash-crowd conditions: 130 leechers join each swarm during the time period between 0 and 10 seconds of our experiment. In both configurations used –native and enhanced *BitTorrent*– leechers join the swarms with no data and remain idle before receiving their first *optimistic unchoke interval*. We refer to this initial time as the *bootstrapping period*. Figures 5(a) and 5(b) show the *bootstrapping period* of the enhanced *BitTorrent* and native *BitTorrent* respectively. The average *bootstrapping period* is 50 seconds under the native and 26 seconds under our enhanced *BitTorrent* protocol. In the native *BitTorrent* (Figure 5(b)), we discern a non-negligible number of leechers experiencing a lengthy *bootstrapping period*, while in the enhanced *BitTorrent* (Figure 5(a)), this metric eases considerably. Leechers with lengthy bootstrapping periods indicate that uploaders of the swarm are unable to locate and unchoke newly-arrived peers. Consequently, the *bootstrapping period* rates depicted in Figure 5(b) reveal that under the native protocol, uploaders are unable to locate leechers demonstrating a low ratio of interest. In contrast, Figure 5(a) shows that under our enhanced protocol all leechers are unchoked early enough to receive data and act as intermediaries. As discussed earlier, a short bootstrapping period is achieved because our unchoking policy (Algorithms 1 and 2) takes into consideration the dynamic situation in which peers find themselves. In addition, Figure 5(b) shows that those leechers that experience a lengthy



(a) enhanced *BitTorrent*



(b) native *BitTorrent*

Fig. 4. *Interested* messages received by seeders under (a) our enhanced and (b) the native *BitTorrent* protocol. In both cases there is an increase in the number of exchanged messages during the first 100 seconds (when peers rapidly join the swarm) and in the long run each seeder receives about 70 *interested* messages.

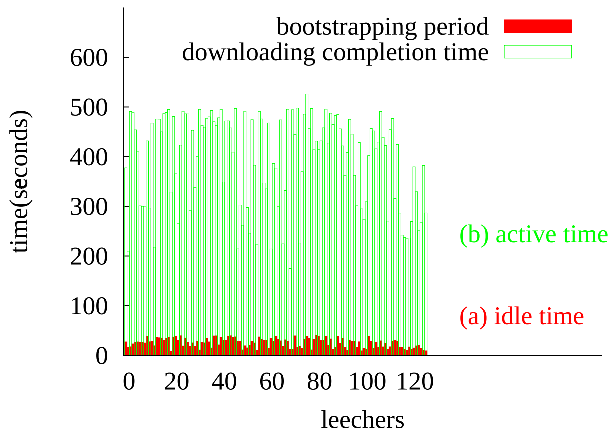
bootstrapping period, also experience a dramatically increased downloading completion time. Under the enhanced *BitTorrent*, all leechers have completed downloading by the 500-th second of the experiment, while under the native protocol, leechers with lengthy *bootstrapping periods* may complete their downloading well past this point in time. Leechers remaining idle for a long time receive no reciprocation and complete their data downloading much later than those who receive reciprocation. Overall, our approach decreases the time that leechers remain idle and increases the period in which leechers act as data intermediaries.

Uploading contribution/Altruism of Leechers

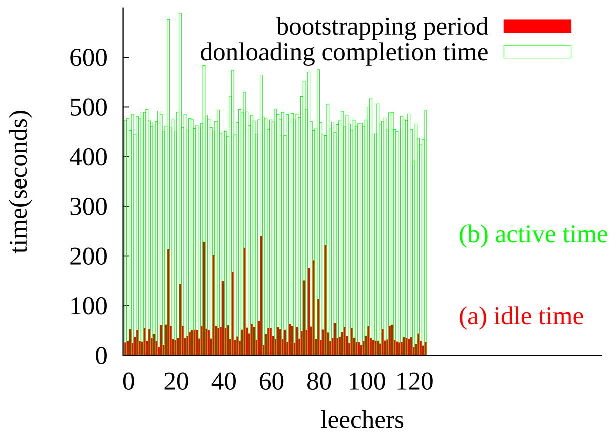
In this section, we compare the uploading contribution of leechers of both protocols. We also examine the *altruism* presented by leechers that we define as the ratio: *pieces uploaded/pieces downloaded*. Figures 6(a) and 6(b) illustrate the number of pieces uploaded as a function of pieces downloaded, and the line $\epsilon : y = x$ which distinguishes between leechers with (i) *altruism* ≥ 1 and (ii) *altruism* ≤ 1. In both cases, we use swarms that consist of 15 seeders and 130 leechers. Leechers join the swarms in flash-crowds and download a fixed number of pieces to obtain a full copy of the distributed file. Although, in the enhanced *BitTorrent* (Figure 6(a)) there is a non-negligible number of peers clustered into area (i), there are only a handful of peers in the same area in the native *BitTorrent* (Figure 6(b)). In the first case, "altruistic" leechers upload more than $2,500$ pieces, but in the second case leechers can upload at most $1,300$ pieces. The leechers found in the area *(i)* act more as uploaders than downloaders. These leechers decongest seeders and provide the swarm with additional uploading capacity of up to $10GB$. As Figure 7 shows, in the native protocol, seeders uploaded $20GB$ of data and leechers uploaded $60GB$ of data. Under the enhanced *BitTorrent*, seeders uploaded $10GB$ and leechers uploaded $70GB$, for the respective experiment. Our approach thus achieves an increase in the contribution of leechers, without involving any complex incentive policy. This is in-line with our key objective to encourage underutilized peers to act as data intermediaries, rather than penalize them. To this end, uploaders unchoke underutilized leechers in an *altruistic* manner. In turn, underutilized leechers obtain data to upload, and ultimately, provide the swarm with additional uploading capacity.

Discussion

We now examine the case in which one or more malicious peers pretend to have a low *ratio of interest* to achieve "bonus" optimistic unchokes. These malicious peers send *have* messages indicating low *ratio of interest* in order to monopolize optimistic unchokes and complete downloading earlier than well-behaved peers, without uploading any data. In the worst case where all peers pretend to have a minimum *ratio of interest*, the enhanced *BitTorrent* degenerates to native *BitTorrent* where the *ratio of interest* simply does not influence optimistic unchoking. To avoid this type of free-riding behavior, we can use a simple token-based policy in which peers exchange data pieces in return for tokens. A token consists of a peer ID, an expiration time after which the token is invalid, a reference to the intended spender of the token, and a file ID referring to the file in distribution. A coordinator (e.g., the tracker) records these four fields when it mints a new token for a particular peer. A token can only be spent by the peer to which it was issued in exchange for blocks of the designated file. Each peer maintains a fixed-size purse of unused tokens issued by the coordinator for use by that peer. Moreover, each peer maintains a ledger of tokens received from other peers in exchange for data blocks. Tokens

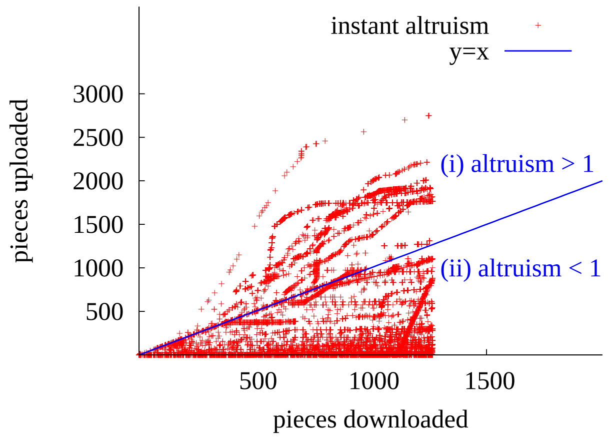


(a) enhanced *BitTorrent*

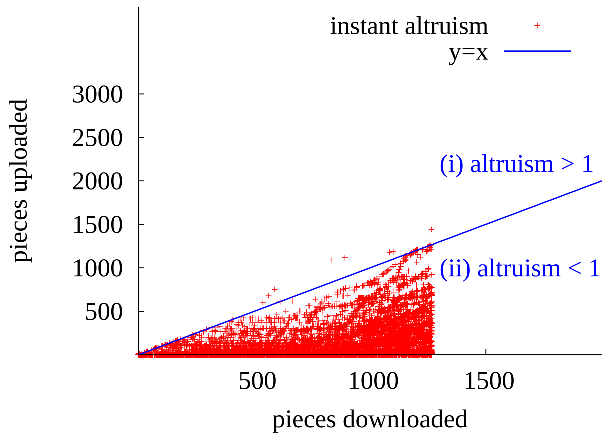


(b) native *BitTorrent*

Fig. 5. Bootstrapping periods and downloading completion times of peers under (a) the enhanced (b) the native *BitTorrent* protocol. In 5(b), a non-negligible number of peers experience a lengthy *bootstrapping period* and an extended downloading completion time. In 5(a), both *bootstrapping period* and downloading completion rates ease considerably.



(a) enhanced *BitTorrent*



(b) native *BitTorrent*

Fig. 6. Altruism presented by leechers under (a) the enhanced and (b) the native *BitTorrent* protocol. In the first case many leechers upload more data than they download (*altruism*$>$ 1). In the second case, leechers display a non-altruistic behavior (*altruism*$<$ 1).

thus flow from the purse of the receiver to the ledger of the sender. Peers communicate periodically with the coordinator to refresh their purses and ledgers. A peer receives a fresh token for its purse in return for each valid token in its ledger. The above technique is introduced in [19] and is applicable to our enhanced *BitTorrent* protocol. Malicious peers that do not upload any data will be unable to refresh their purses with fresh tokens, and consequently will remain idle until they upload obtained data blocks. Another scheme to prevent free-riding in our enhanced *BitTorrent* protocol is the one proposed in [22]. File pieces are encrypted and leechers could barter with each other by exchanging decryption subkeys for file pieces. A peer must upload an intact encrypted data piece before receiving a decryption subkey. This is referred to as the "data first, key later" (DFKL) rule. No centralized authority is required. This scheme is called "treat-before-trick" (TBeT) and penalizes free-riders with increased file completion times (time to download file and necessary subkeys to decrypt file pieces).

VI. Conclusion

In this paper, we present the enhanced *BitTorrent* protocol whose unchoking policy better harnesses underutilized peers that have few clients interested in downloading data from them. Our proposal involves uploaders allocating optimistic

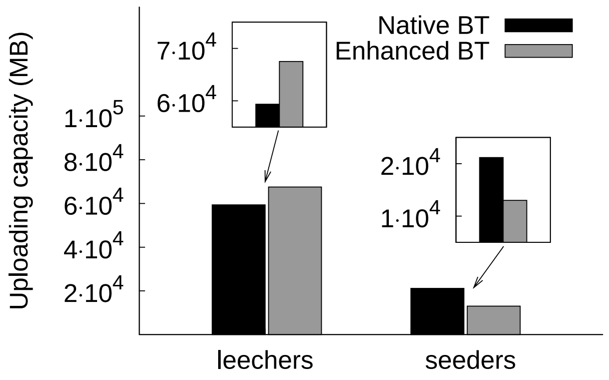


Fig. 7. Aggregate uploading contribution of leechers and seeders under the enhanced and the native *BitTorrent* protocol. Under the enhanced *BitTorrent* protocol leechers upload 68 GB of data and under the native *BitTorrent* protocol leechers upload $58GB$ of data.

unchoking slots to underutilized peers. This policy enables peers to obtain data and essentially act as content intermediaries, rather than remain idle. Experimentation with leecher and tracker prototypes shows that our approach achieves improved quality of inter-connection amongst peers compared with the native *BitTorrent* protocol. Under our enhanced *BitTorrent* protocol, the number of directly-connected and interested-in-cooperation peers increases significantly. A substantial portion of the peers in question act as data intermediaries and consequently, better peer content distribution is achieved. Moreover, our modified *BitTorrent* protocol has the effect of creating altruistic leechers who act more as uploaders than downloaders. The net result is that these altruistic leechers furnish uploading capacity that helps relieve the burden of seeders. Finally, unlike prior works aiming to improve the performance of the native *BitTorrent* protocol, our enhanced *BitTorrent* achieves a shorter bootstrapping period and a shorter downloading time, *without* the use of complex incentive policies.

Acknowledgments: We would like to thank Y. Mimiyannis, Y. Kamonas, C. Christou and A. Sevastidou for their help during our experimentation. This work was partially funded by the *D4SII EU-FP6* project.

References

[1] The bittorrent protocol. http://www.bittorrent.org/beps.
[2] Coolstream. http://www.coolstreaming.us/.
[3] Fasttrack. http://www.kazaa.com/.
[4] Gnutella. http://gtk-gnutella.sourceforge.net/.
[5] Internet study. http://www.ipoque.com/en/resources/internet-studies.
[6] Mininova. http://www.mininova.org/.
[7] Napster. http://music.napster.com/.
[8] Pplive. http://www.pplive.com.
[9] Skype. http://www.skype.org/intl/en/home.
[10] Azureus(vuze). http://azureus.sourceforge.net/, January 2011.
[11] Alix L. H. Chow, L. Golubchik, and V. Misra. Bittorrent: An extensible heterogeneous model. In *INFOCOM*, pages 585–593, Rio De Janeiro, Brazil, April 2009.
[12] B. Cohen. Incentives build robustness in bittorrent. In *IPTPS*, Berkeley, CA, USA, February 2003.
[13] J.R. Jiang, J.S. Chiou, and S.Y. Hu. Enhancing neighborship consistency for peer-to-peer distributed virtual environments. In *ICDCS Workshops*, Toronto, Canada, June 2007.
[14] R. Landa, D. Griffin, R.G. Clegg, E. Mykoniati, and M. Rio. A sybilproof indirect reciprocity mechanism for peer-to-peer networks. In *INFOCOM*, pages 343–351, Rio De Janeiro, Brazil, April 2009.
[15] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in bittorrent systems. In *SIGMETRICS*, pages 301–312, San Diego, CA, USA, June 2007.
[16] D.S. Menasché, L. Massoulié, and D.F. Towsley. Reciprocity and barter in peer-to-peer systems. In *INFOCOM*, San Diego, CA, USA, March 2010.
[17] M. Meulpolder, Dick H.J. Epema, and H.J. Sips. Replication in bandwidth-symmetric bittorrent networks. In *International Parallel and Distributed Processing Symposium/International Parallel Processing Symposium*, pages 1–8, Miami, Florida, USA, April 2008.
[18] M.Y. and Y. Yang. An efficient hybrid peer-to-peer system for distributed data sharing. *IEEE Transactions on Computers*, 59(9):1158–1171, September 2010.
[19] R. Peterson and E.G. Sirer. Antfarm: Efficient content distribution with managed swarms. In *NSDI*, pages 107–122, Massachusetts, Boston, USA, April 2009.
[20] M. Piatek, T. Isdal, T.E. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent? In *4th USENIX Symposium on Networked Systems Design & Implementation*, Cambridge, MA, April 2007.
[21] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang. Topbt: A topology-aware and infrastructure-independent bittorrent client. In *INFOCOM*, pages 1523–1531, San Diego, CA, USA, March 2010.
[22] K. Shin, D.S. Reeves, and I. Rhee. Treat-before-trick: Free-riding prevention for bittorrent-like peer-to-peer networks. In *23rd IEEE Int. Symposium on Parallel and Distributed Processing (IPDPS'09)*, pages 1–12, Rome, Italy, May 2009.
[23] D. K. Vassilakis and V. Vassalos. An analysis of peer-to-peer networks with altruistic peers. *Peer-to-Peer Networking and Applications*, 2(2):109–127, June 2009.
[24] D.I. Wolinsky, P.S. Juste, P.O. Boykin, and Renato J. O. Figueiredo. Addressing the p2p bootstrap problem for small overlay networks. In *Peer-to-Peer Computing*, pages 1–10, Delft, The Netherlands, August 2010.
[25] M. Yang, Q. Feng, Y. Dai, and Z. Zhang. A multi-dimensional reputation system combined with trust and incentive mechanisms in p2p file sharing systems. In *ICDCS Workshops*, Toronto, Canada, June 2007.