EnhancedBit: Unleashing the Potential of the Unchoking Policy in the *BitTorrent* Protocol

V. Atlidakis[a,*], M. Roussopoulos[b], A. Delis[b]

[a]*European Organization for Nuclear Research (CERN), IT Departement, CH-1211 Genève 23, Switzerland*
[b]*University of Athens, Department of informatics and Telecommunications, GR-15784 Ilisia, Athens, Greece*

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, i.e., increases the number of directly-connected and interested-in-cooperation peers without penalizing underutilized and/or idle peers. Our *optimistic unchoking policy* takes into consideration the number of clients currently interested in downloading from a peer that is to be unchoked. Our conjecture is that peers having few clients interested in downloading data from them, should be favored with optimistic unchoke intervals. This enables the peers in question to receive data since they become unchoked faster and in turn, they will trigger the interest of additional clients. In contrast, peers with plenty of "interested" clients should enjoy a lower priority to be selected as *planned optimistic unchoked*, since these peers likely have enough data to forward; nevertheless, they receive enough data due to tit-for-tat peer reciprocation and are not in need of *optimistic unchoking* slots. Armed with this realization, we establish an analytical model and prove a significant performance improvement under our modified *BitTorrent* protocol. Experimental results, also, indicate that our approach significantly outperforms the existing *optimistic unchoking policy* in three important aspects: firstly, there is a higher number of interested-in-cooperation and directly-connected peers. Secondly, since leechers now act as data intermediaries, the load on seeders eases up considerably. Last, a shorter bootstrapping period for fresh peers is achieved. Hence, we claim that our approach helps implement an enhanced *BitTorrent* protocol and we name it "*EnhancedBit*".

Keywords: Peer-to-peer (P2P), Content Distribution, Incentive Protocols, BitTorrent

---

*Corresponding author
Email addresses: `v.atlidakis@gmail.com` (V. Atlidakis), `mema@di.uoa.gr` (M. Roussopoulos), `ad@di.uoa.gr` (A. Delis)

1. Introduction

Peer-to-peer applications remain of crucial importance as there is still a growing trend for exchange of large multimedia files, voice over IP and broadcasting of TV-quality programs 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 do not distinguish between the roles 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.

BitTorrent protocol [5] is the most popular *peer-to-peer* protocol for bulk data transfer [9, 23, 22], accounting for approximately 27-55% of all Internet traffic depending on geographical location [2]. In the pre-*BitTorrent* era, Napster, Gnutella and Kazaa were widely-used protocols for transferring multimedia files, such as mp3's, movies, and software. However, their centralized indexing methods [4] and/or the lack of a tit-for-tat schema among peers prevented them from being an effective competitor to *BitTorrent*'s dominance.

As shown in Figure 1, the *BitTorrent* protocol operates in three different layers: At the *swarm layer* [22], a peer contacts a tracker to join a swarm and receive a list of other peers to connect to. 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 [9]. 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 on the *neighborhood layer* and modify the neighborhood selection mechanism of the protocol known as *peer unchoking*; this includes *regular unchoking* [9, 23] and *optimistic unchoking* [8, 19]. *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* intervals to downloaders to enhance the cooperation of peers. The existing *optimistic unchoking policy* [9] uses a round-robin approach giving priority to more recently connected peers. This approach guarantees at least one bootstrapping interval for any new peer, regardless of the situation (i.e., dynamics) 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

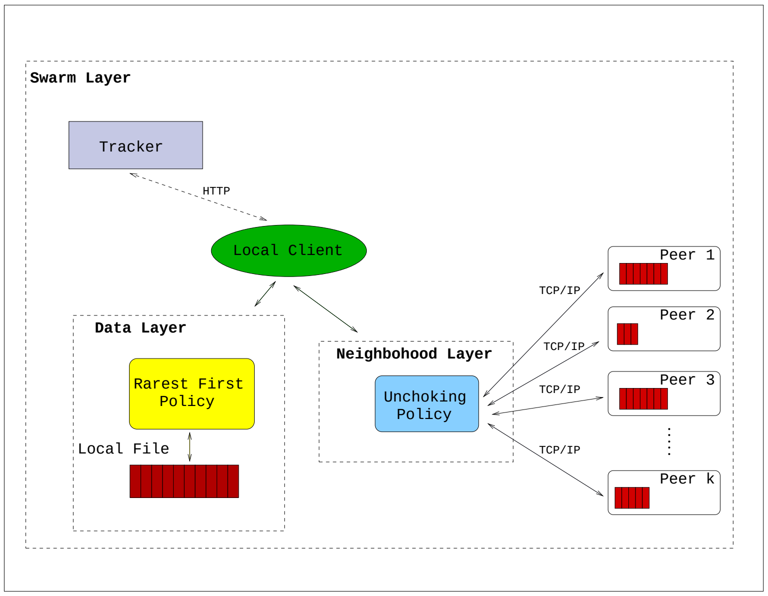


Figure 1: High level view of the *BitTorrent* protocol.

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 peers having few clients interested in downloading data, should be favored with *optimistic unchoke* intervals. In turn, this approach enables the peers in discussion to receive data since they get unchoked and so, they may trigger the interest of additional clients. To this end, every time an *optimistic unchoking* should be performed, the peer with the least number of *interested* connections is selected as *planned optimistic unchoked*. Uploading peers with few clients 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. These include: the number of *interested* connections maintained by peers during data dissemination, the bootstrapping period of new peers and downloading time, the correlation between *unchoke*, *interested* messages and data uploaded by leechers, the *altruism* presented by leechers and finally, the uploading contribution of leechers against seeders. The contributions of our work are:

1. Enhancement of the *BitTorrent* protocol that improves the quality of inter-connection of peers, i.e., increases the number of directly connected and

interested-in-cooperation peers.

2. Decongestion of seeders as more leechers now act as data intermediaries and so the load on seeders eases up considerably.
3. A shorter bootstrapping period for fresh peers compared with the native *BitTorrent* protocol.

Prior related research has been carried out in a number of aspects including reciprocity mechanisms [20, 13], tit-for-tat schemas to discourage free riding [27, 22], and incentives policies in [23, 17]. These efforts have suggested solutions regarding the modification of the *regular unchoking policy*, and have introduced techniques to encourage peers to act as uploaders and to discard idle peers. Our work, however, is a new method to modify the *optimistic unchoking policy* in order to encourage cooperation of peers. A notably similar approach to ours is presented by the authors of [19], where a modified optimistic unchoking policy to prevent free-riding is suggested. Under this approach, uploaders entitle optimistic unchoking slots to peers based on past behaviour: a gain-value parameter is calculated for each peer that has received optimistic unchoking slots, and the peer with the highest gain-value will receive the upcoming optimistic unchoking slot. As oppposted to [19], in *EnhancedBit* our objective is to treat underutilized peers 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. The suggested *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.

The rest of the work is organized as follows. Section 2 briefly presents prior work related to *BitTorrent* and Section 3 provides necessary background on the *BitTorrent* protocol. Section 4, outlines the key features of our proposed *optimistic unchoking* scheme and Section 5 introduces an analytical model to capture the performance improvement under our *EnhancedBit* protocol. Section 6 presents our main experimental results and Section 7 offers our concluding remarks.

2. Related Work

After the seminal publication of the *BitTorrent* protocol [9], a plethora of analytical models have been built to study the performance characteristics of *BitTorrent*-like networks. Downloading completion time and effectiveness of file sharing, in steady state *BitTorrent* swarms, have initially been examined by Qiu et al. [24]. Downloading time and downloading rate of peers are modeled in environments with heterogeneous users (in terms of bandwidth) in [18, 12]. *BitTorrent* is analyzed under a game-theoretic perspective in [25], where downloading completion time of peers, protocol's robustness and performance are evaluated. Furthermore, in [30] the use of altruism in *P2P* networks is examined; altruism is defined via a parameter that reflects the 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. Neighborship consistency is defined by Jiang et al. [14] 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 of connectivity in *P2P* systems. In our *EnhancedBit* protocol we define the *ratio of interest* of peers (Section 4.2) and the *benefit obtained* by peers (Section 5.3) during data dissemination. In [19], the authors propose an optimistic unchoking method to prevent free-riding during optimistic unchoking slots. To achive this, peers maintain history information regarding uploading contribution of neighboors that have received optimistic unchokes. Based on past behaviour a gain-value parameter is calculated per neighboor, and each peer entitles optimistic unchoking slots to the neighbor yielding the highest gain-value. Under our approach, a peer decides which counterpart to unchoke in order to maximize the counterpart's 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.

Regular unchoking policy tit-for-tat schema is thoroughly examined in a large body of work: [20, 23, 28, 17, 27, 31, 22, 15]. The primary question is whether it preserves a rational uploading and downloading reciprocation of peers, or not. In [29] an innovative peer selection strategy is proposed. Peers are modelled depending on download completedness. Also, file availability and the dying process of the system is analyzed. The innovative peer selection strategy suggested enables more peers to finish the download job and prolongs the systems lifetime by alleviating its dying process. In [20, 13], indirect and direct reciprocity mechanisms are examined. Peers may exploit their own data contributions to obtain data from others (direct reciprocation), or may contribute resources to one set of peers and use this contribution to obtain services from a different set of peers (indirect reciprocation). Our approach differs from the above efforts as we suggest an unchoking policy in which peers altruistically offer data to underutilized and/or idle counterparts. In [23, 28] the issue of incentive compatibility is re-examined. The authors show 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. Incentives in *BitTorrent* systems are also studied in [17], where the unchoking algorithm of the 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 *EnhancedBit* protocol. A variety of mechanisms for preventing free-riding in *P2P* file-sharing systems have been applied in [27, 31, 22, 15]. Although applying mechanisms to discourage free-riding is essential to steering more peers to act as data intermediaries, it does not address the problem of bootstrapping idle peers with no data to upload. With our improved unchoking policy, uploaders immediately locate and furnish data to peers with no initial data blocks. Moreover, the *piece selection strategy* of the original *BitTorrent* protocol is examined in recent work [11], where an Interest-Intend Piece Pelection (IIPS) is suggested. Every IIPS peer selects to download pieces that, if downloaded, would enhanced the cooperation of peers. In our schema, every *EnhancedBit* peer allocates its *optimistic unchoke intervals* to enhance the co-operation of peers.

There have been many proposals for new peer-to-peer content distribution protocols, hybrid peer-to-peer data sharig systems, or improved *BitTorrent*-like protocols: [26, 21, 22, 16, 27]. However, our work suggests 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.

3. Background

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

3.1. Terminology

Files transferred 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. A peer may be a downloader and/or an uploader. 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. A *swarm* consists of *seeders* and *leechers* that are sharing a specific file. Yet, in respect to a popular file there may be plenty of independent and geographically distributed *swarms*.

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 *.torrent* websites acting as global directories of available files. Torrent sites are either public (e.g., Isohunt [3]), or private, known as *Darknets* [32] (e.g., TorrentLeech [6]).

3.2. 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* [7] that uses *DHT*s.

3.3. 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*. Every time the algorithm is invoked the local host sorts peers having sent him data according to their uploading rate 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. In the introducing paper of the *BitTorrent* protocol [9] three *regular unchoked* and one *optimistic unchoked* peers were used. In later works different settings were used: In [27, 18, 10, 24] four *regular unchoked* and one *optimistic unchoked* were used, while in [12] four *regular unchoked* and two *optimistic unchoked* peers were used.

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.

3.4. 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:

- **choked**: The local 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.
- **having-an-interest**: The local client as a downloader is interested in receiving pieces from the remote peer (uploader).

- **choking**: The local client as an uploader chokes the remote peer (downloader). The client must discard any unanswered requests of the remote peer.
- **interested**: The remote peer (downloader) is interested in receiving pieces from the local client (uploader).

3.5. 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, as shown in recent works (e.g., [11]), completion of downloading may be delayed by a last missing piece.

3.6. Overview of BitTorrent *Operation*

As shown in Figure 2, at first, a peer downloads a *.torrent* metadata file to contact a tracker and receive a set of 50 peers with whom to establish *TCP* connections. After joining the swarm, Figure 3, the peer with no data to upload is interested in downloading data from every participant in the 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, depicted in Figure 4, 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 than 40 initiated connections, downloads from all of them but uploads at most to 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.

[1]three regular unchoked and one optimistic unchoked

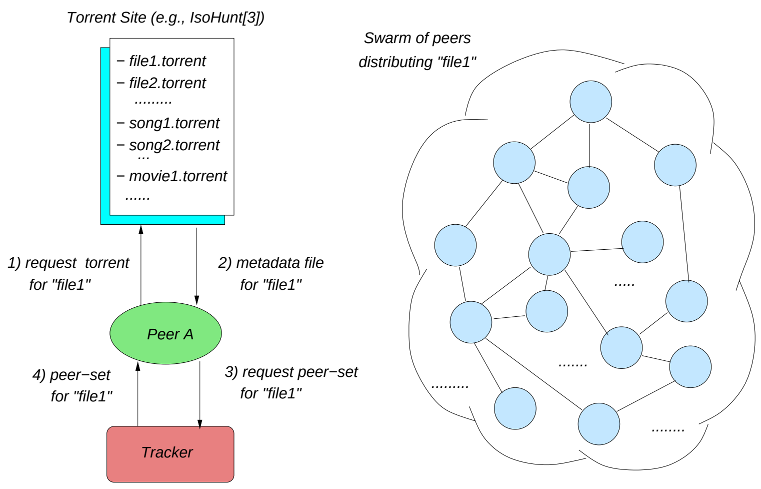

Torrent Site (e.g., IsoHunt[3])
– file1.torrent
– file2.torrent
– song1.torrent
– song2.torrent
– movie1.torrent
1) request torrent for "file1"
2) metadata file for "file1"
Peer A
4) peer–set for "file1"
3) request peer–set for "file1"
Tracker
Swarm of peers distributing "file1"

Figure 2: Peer A requests *.torrent* metadata file for "file1" to join the swarm.

4. *EnhancedBit*

As discussed earlier, under the *peer unchoking policy* of the original *BitTorrent* protocol a peer is selected at random to receive *optimistic unchoking* slots. To modify the random selection of the native *BitTorrent* protocol, we introduce an approach under which peers allocate *optimistic unchoking* slots to underutilized counterparts, selectively. In turn, initially underutilized peers will receive data and will act as data intermediaries, rather than remaining idle. In what follows, we outline our proposed *peer unchoking policy* by first introducing the messages used by our *EnhancedBit* protocol. We then discuss the metrics used for maximizing the ratio of interest and the algorithms used for our *unchoking policy.*

4.1. EnhancedBit *Messages*

To implement our *EnhancedBit* protocol we use the messages of the original *BitTorrent* protocol, but we augment the *have state-oriented* message with an additional float value. The latter corresponds to the *ratio of interest* (Section 4.2) of the sender of the *have* message and helps us implement our enhanced *unchoking policy.* The messages in use can be categorized into: *swarm-oriented, state-oriented* and *data-oriented* messages.

Table 1 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 2. We refer to these as *state-oriented* messages that help achieve cooperation among

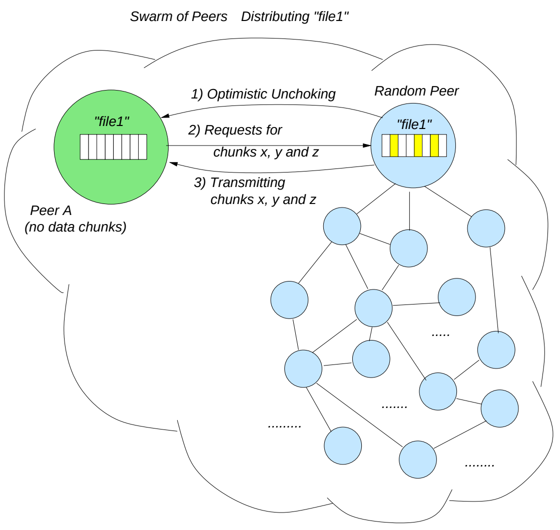

Swarm of Peers Distributing "file1"
1) Optimistic Unchoking
Random Peer
"file1"
2) Requests for chunks x, y and z
"file1"
3) Transmitting chunks x, y and z
Peer A
(no data chunks)

Figure 3: Peer A receives its first data chunks during the first *optimistic unchoke interval*.

peers and implement the *peer unchoking policy*. All messages of Table 2 contain no downloadable data but designate when peers must exchange data or not (Section 3.4). 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 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. Finally, Table 3 summarizes the *data-oriented* messages that are sent between *unchoked* peers (i.e., peers that are exchanging data).

4.2. Peer Unchoking – Ratio of Interest

We define the *ratio of interest* RI_p of a peer p to be $\mathbf{RI_p} = \mathbf{int_p}/\nu$, where int_p is the number of *interested connections* p maintains, and ν is the number of clients remotely connected to peer p. Under a specific *BitTorrent* swarm all peers maintain the same number of remote connected clients, usually fixed to 40. In contrast, the number of *interested connections* maintained by a peer varies and may help project the amount of data requests the peer in question will ultimately receive provided that data requests are received only via connections marked 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. We insert the *ratio of interest* (RI_p) as the measure that reflects a peer's uploading utilization. The key objective of the *unchoking policy* of our *EnhancedBit* protocol is to maximize the *ratio of interest* of peers. We anticipate that our *unchoking policy*

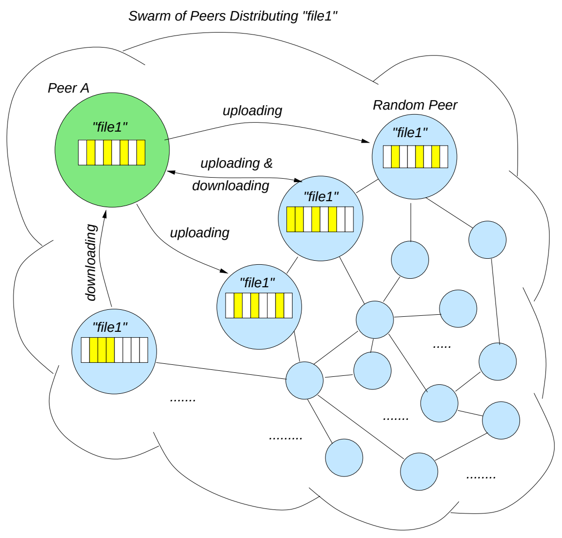


Figure 4: Peer A uploads and/or downloads data chunks from multiple peers.

will be most effective when we rotate the *planned optimistic unchoked* peer in a prioritized way, yielding the right-of-way to fresh peers and peers with minimum interest ratios. 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. 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.

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, i.e., an increase in the number of directly-connected and interested-in-cooperation peers. An improvement in the quality of inter-connections is attained as soon as underutilized peers are effectively located and provided with data to disseminate in swarm. In turn, underutilized peers will act as data intermediaries, instead of remaining idle. The peers that have saturated their uplinks will be decongested as more clients will act as content intermediaries. In Section 5, we present an analytical model to compare the unchoking policies of our *EnhancedBit* and the native *BitTorrent* protocol. Also, in Section 6, we experiment with clients implementing *EnhancedBit* and the original *unchoking schema*.

4.3. Algorithms

Our *EnhancedBit* protocol invokes Algorithms 1 and 2 when local 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 [5]. As

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 1: Swarm-oriented Messages

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 *Downloaders* of the local client, the set of remote *Uploaders* 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. Although *uninterested* peers may be selected unchoked until an *interested* peer is added to the RU set, 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 *Downloaders* of 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 the first 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).

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.
uninterested: Peer A sends this message to remote peer B when A is not interested in receiving data from B, whatsoever; 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 a connection with peer B. Payload includes file identifier and peer identifier of peer A.

Table 2: State-oriented Messages

5. Analytical Model

In what follows we present a probabilistic mathematical model that incorporates basic system parameters of *BitTorrent*-like content sharing protocols. We form expressions that encapsulate delicate performance characteristics of *BitTorrent*-like protocols and compare the *ratio of interest* of peers under our *EnhancedBit* and the native *BitTorrent* protocol. In addition, we compare the bootstrapping period of peers in the systems under discussion. Our purpose is to help the reader obtain an analytical perspective on how our *EnhancedBit* protocol differs from the native *BitTorrent* protocol. We analyze a swarm S

request: The sender of this message includes three 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, the block's offset within piece and the 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 three integers indicating piece index, block offset, and block length.

Table 3: Data-oriented Messages

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

```
Input: Uploaders, Downloaders, $RI_{p \in Downloaders}$
1: $Interested \leftarrow \{p : \forall p \in$ Downloaders $AND$ $p$ interested in local client$\}$
2: if $round = 1$ then
3:     OU $\leftarrow \{p : Min\{RI_p\} \forall p \in Interested\}$
4:     unchoke OU
5: end if
6: RU $\leftarrow \{p : \forall p \in Top3\ Uploaders\}$
7: for $p \in Interested$ do
8:     if $p \in RU$ then
9:         unchoke p
10:    else
11:        choke p
12:    end if
13: end for
14: if $OU \subseteq RU$ then
15:    repeat
16:        choose $p \in Downloaders$
17:        unchoke $p$
18:    until $p \in Interested$
19: end if
```

composed of N cooperating peers, in steady state. We assume that N_l peers are leechers and s peers are seeders. At any time a peer p possesses $n_p \leq n$ pieces of F, while a seeder maintains a complete file copy of n data pieces. Every peer as an uploader may receive data requests from ν remote connected downloaders. Under these assumptions, we present our mathematical model as follows.

5.1. Analyzing Data Replication

Bernoulli Trial

We pick one downloader j from a random peer i with ν remote connected downloaders. We consider "success" the event that j needs a piece from i and "failure" the event that j does not need any piece from i. Let p be the probability of "success" in the above Bernoulli trial [1] and q the probability of "failure".

Binomial experiment

We repeat the above trial for each of the ν downloaders of peer j. These trials are ν independent Bernoulli trials, and consequently follow the binomial distribution [1]. We insert the random variable X that expresses the number of interested downloaders of peer j. The variable X follows the binomial distribution with probability mass function $f : \{0, 1, ..., \nu\} \to [0, 1]$ that is:

$$f(x) = P(x = X) = \binom{\nu}{x} p^x (1-p)^{\nu - x} \tag{1}$$

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

```
Input: $Downloaders, RI_{p \in Downloaders}$
1: $temp1 \leftarrow \{p : \forall p \in$ Downloaders $AND$ has pending requests $OR$ recently unchoked $\}$
2: sort $temp1$ according to last unchoke time
3: $temp2 \leftarrow \{p : \forall p \in$ Downloaders $AND$ p $\notin$ $temp1\}$
4: sort $temp2$ according to downloading rate
5: if $round = 1, 2$ then
6:     $RU \leftarrow \{p_{i=1,2,3} \in temp1 + temp2\}$
7:     $OU \leftarrow \{p : Min\{RI_p\} \forall p \in temp1 + temp2\}$
8:     unchoke OU
9: else
10:    $RU \leftarrow \{p_{i=1,2,3,4} \in temp1 + temp2\}$
11: end if
12: for $p \in D$ do
13:    if $p \in RU$ then
14:        unchoke p
15:    else
16:        choke p
17:    end if
18: end for
```

The cumulative distribution function is:

$$F(x) = P(X \leq x) = \sum_{k=1}^{x} f(k) \tag{2}$$

mean value μ is:

$$\mu_X = \nu \cdot p \tag{3}$$

and variance σ^2 is:

$$\sigma_X^2 = \nu \cdot p \cdot q \tag{4}$$

If we divide the above equations with the fixed number of remote connected downloaders (ν), we get the mean value and the variance for the *ratio of interest*; these are: $\mu_{RI} = p$ and $\sigma_{RI}^2 = p \cdot q$. In the upcoming evaluation section, with equations (3) and (4) we will calculate p and q in real swarms.

5.2. Analyzing Data Flow

We select a peer p with int_p *interested connections* and examine the flow of a random piece x_i to p. Before receiving piece x_i, peer p maintains int_p *interested connections*. After receiving piece x_i, peer p maintains int'_p *interested connections*. We insert the sampling space $\Omega = \{(int_p, int'_p) : int_p, int'_p = 0, 1, 2, ..., \nu\}$ with sample space size $N(\Omega) = (\nu + 1)^2$.

Definition We define the event INC_p that peer p obtains a higher number of *interested connections* upon receiving piece x_i, and symbolize: $INC_p = \{(int_p, int'_p) \in \Omega : int'_p > int_p\}$.

It is $N(INC_p) = \nu - int_p$ and $N(\Omega) = (\nu+1)^2$; the probability that the event INC_p occurs, is:

$$P(INC_p) = \frac{\nu - int_p}{(\nu+1)^2} \tag{5}$$

Assume two random peers p_1, p_2 with $P(INC_{p_1}) > P(INC_{p_2})$, from equation (5) we get:

$$\frac{\nu - int_{p_1}}{(\nu+1)^2} > \frac{\nu - int_{p_2}}{(\nu+1)^2} \Leftrightarrow$$
$$int_{p_1} < int_{p_2}$$

We can express the following Lemma:

Lemma 1. *For any two peers p_1, p_2 it is $P(INC_{p_1}) > P(INC_{p_2})$ if and only if $int_{p_1} < int_{p_2}$.*

Lemma 1 expresses what is intuitively clear: A data piece x_i should be uploaded to the peer with the least number of *interested connections* (or equivalent the lowest *ratio of interest*) to achieve the highest increase in the number of *interested connections*. As presented in Sections 4.2 and 4.3, under our *Enhanced-Bit* uploaders select as *planned optimistic unchoked* the peer with the lowest RI_p. This is done in an attempt to improve the quality of inter-connection of peers, i.e., increase the number of *interested connections* maintained amongst cooperating peers.

5.3. Benefit Obtained

We define the instant benefit obtained (b_{p,x_i}) in peer p, as the number of the additional *interested connections* that triggered the flow of piece x_i to p. It is

$$b_{p,x_i} = int'_{p,x_i} - int_p \tag{6}$$

where int_p is the number of *interested connections* in p before receiving piece x_i, and int'_{p,x_i} the number of *interested connections* after receiving piece x_i. If $b_{p,x_i} < 0$, instead of instant benefit, there is an instant loss in the number of *interested connections.*

The local benefit obtained in peer p after downloading n pieces is:

$$B_p = \sum_{i=0}^{n} (int'_{p,x_i} - int_p). \tag{7}$$

Local benefit $B_p = \sum\limits_{i=0}^{n} b_{p,x_i}$ measures the number of additional downloaders that became interested in downloading data from p, when p receives pieces $\{x_0, x_1, ..., x_n\}$.

The aggregate benefit obtained in swarm is: $B = \sum_{p \in S} B_p$. If there are N_l leechers in swarm that downloaded $\{n_1, n_2, ..., n_{N_l}\}$ data pieces respectively, the aggregate benefit in swarm is:

$$B = \sum_{p \in S} B_p = \sum_{i=1}^{N_l} \sum_{j=0}^{n_i} (int'_{p,x_j} - int_p) \tag{8}$$

At this point we make a conservative assumption, i.e., every peer p obtains only one more *interested connection* after receiving piece x_i. This means that the flow of every piece x_i to a peer p only triggers the interest of one additional downloader. After the flow of n_i random pieces to p, the local benefit will be:

$$B_p = n_i \cdot P(INC_p)$$

From (8) we get:

$$B = \sum_{i=1}^{N_l} n_i \cdot P(INC_{p_i}) \tag{9}$$

With the above, we are at a position to compare aggregate benefit obtained under the *EnhancedBit* and the native *BitTorrent* protocol.

Proposition 1. *For the aggregate benefit attained under the* EnhancedBit *and the native* BitTorrent *protocol it stands:* $B_{EN\_BT} > B_{BT}$.

Proof. Recall that under the *unchoking policy* of the *EnhancedBit* with algorithms 1 and 2, we select as *planned optimistic unchoked* the peer p with the minimum *ratio of interest*. Under the unbiased selection of the native *BitTorrent* a random peer p' is selected as *planned optimistic unchoked*. Obviously, $int_p \leq int_{p'}$ and from Lemma 1 it comes out that $P(INC_{p,x_i}) > P(INC_{p',x_i})$ or equivalent: $P_{EN\_BT}(INC_{p,x_i}) > P_{BT}(INC_{p,x_i})$. We can extend our observation for $\{n_1, n_2, ..., n_{N_l}\}$ random pieces that flow to a cluster of leechers $\{1, 2, ..., N_l\}$. We can write: $\sum_{i=1}^{N_l} n_i \cdot P_{EN\_BT}(INC_{p_i}) > \sum_{i=1}^{N_l} n_i \cdot P_{BT}(INC_{p_i}) \Leftrightarrow$ $\mathbf{B_{EN\_BT}} > \mathbf{B_{BT}}$.

$\square$

Proposition 1 reflects the fact that the flow of pieces under the *EnhancedBit* will provoke an increased *ratio of interest* compared to native *BitTorrent*. This means that during data dissemination peers of the *EnhancedBit* protocol will maintain a higher number of *interested connections* and will ultimately receive more data requests. Due to the increased quality of inter-connection peers are probe to act as data intermediaries, instead of remaining idle.

5.4. Bootstrapping Period

As explained in Section 3, every 30 seconds the local peer selects as *planned optimistic unchoked* one client of its ν remote connected downloaders. The downloader selected as *planned optimistic unchoked* will be kept unchoked for one optmistic unchoke interval (30 seconds), regardless of its contribution to local peer. *Optimistic unchoking schema* guarantees that a new client will be able to download at least one piece without having sent any. This is done in an attempt to locate and bootstrap "fresh" clients with no data, whatsoever. In a period of time as much as 30 seconds, the probability that a downloader will be selected as *planned optimistic unchoked* is:

$$P = 1 - (\frac{\nu - 1}{\nu})^{\nu} \tag{10}$$

The above formula stands under the assumption that the client under examination is connected to ν uploaders for at least 30 seconds. These ν uploaders will unchoke ν random downloaders, after a maximum time of 30 seconds. The probability that the downloader in discussion will receive none of these ν *optimistic unchoking* slots is $P' = (\frac{\nu-1}{\nu})^{\nu}$. As opposed to P', the probability that the downloader in discussion will receive at least one *optimistic unchoking* slot is: $P = 1 - (\frac{\nu-1}{\nu})^{\nu}$. Equation (10) gives us $P = 0.637$ for $\nu = 40$. This means that an underutilized or a "fresh" peer will be located from the *unchoking policy* of the original *BitTorrent* with a probability 0.637 (or 63.7%) after 30 seconds.

Under the *unchoking policy* of the *EnhancedBit* (algorithms 1, 2) *optimistic unchoking* slots are distributed to the cluster of peers with low *ratio of interest*, i.e. new or underutilized peers. If there are N_u peers with low *ratio of interest*, each uploader is connected with a fraction of $\frac{N_u}{N}$ of underutilized peers. [2] In the same spirit, the probability that an underutilized peer will receive at least one *optimistic unchoking* slot is:

$$P = 1 - (\frac{\nu - 1}{\nu} \cdot \frac{N_u}{N})^{\nu} = 1 - (\frac{\nu - 1}{\nu})^{\nu} \cdot (\frac{N_u}{N})^{\nu}$$

which gives us $P = 1$, since $(\frac{N_u}{N})^{\nu} \to 0$. This means that an underutilized or a "fresh" peer will be located from the *unchoking policy* of the *EnhancedBit* with a probability 1 (or 100%) after 30 seconds.

From the above analysis we conclude that the *unchoking schema* of the *EnhancedBit* is 57% more effective than the *unchoking schema* of the native *BitTorrent* in locating and bootstrapping an underutilized peer. As a result, peers experience a shorter bootstrapping period under our *EnhancedBit* system. The experiments presented in Section 6 confirm that an improvement in the bootstrapping period of peers is indeed attained under our *EnhancedBit* system.

6. Evaluation

To evaluate our *EnhancedBit* protocol, we have implemented in *Python* a respective client as well as a tracker. Our implementation of both the client

[2] underutilized peers are uniformly distributed amongst N peers

and the tracker run in *Windows7*, *Linux*, and *MacOS*. For our experiments, we used 40 workstations, each featuring a $1GHz$ clock and $1GB$ memory running *GNU/Linux*. The workstations are attached to a local Ethernet network running at $100Mps$. 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 *EnhancedBit* and the native *BitTorrent*.
2. Compare the bootstrapping period and the downloading completion time achieved by leechers under the *EnhancedBit* and the native *BitTorrent* protocols.
3. Examine the relation between *unchoke* and *interested* messages received by leechers, as well as, data uploaded by leechers under the *EnhancedBit* and the native *BitTorrent* protocols.
4. Analyze the pieces uploaded from leechers and seeders to ascertain the degree of altruism presented by *EnhancedBit* leechers, and to evaluate the decongestion of seeders achieved by our *EnhancedBit* approach.

We experimented with numerous settings regarding the distribution of seeders and leechers among our clients. We used a number of key parameters that we outline in Table 4. 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. We emphasize that the swarms under examination were not in steady state: Leechers departed swarms as soon as they downloaded a complete file copy.

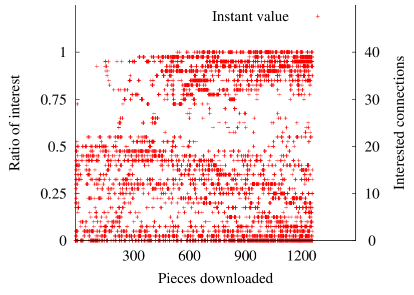
Optimistic unchoke interval	30 seconds
Regular unchoke interval	10 seconds
Regular unchoked peers	3
Optimistic unchoked peers	1
Peer-set size	40
Swarm	120 leechers, 15 seeders
Data pieces in file	1267
Piece size	$512KB$
File size	$\approx 630MB$

Table 4: Experimental Parameters

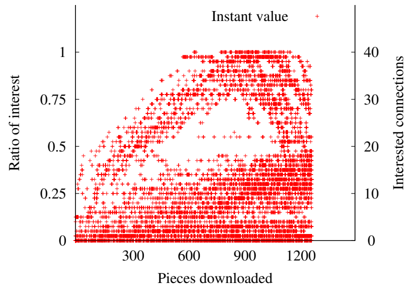
Experiment 1: Ratio of Interest

First, we examine the *ratio of interest* of peers, as defined in Section 4. 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 uploading utilization of peers. We examine *ratio of interest* as a function of downloaded pieces (Figures 5, 6), and as a function of downloading completion time (Figure 7). In both settings used –native and

EnhancedBit– swarms are formed from 120 leechers and 15 seeders; leechers join swarm at random times within 100 seconds.

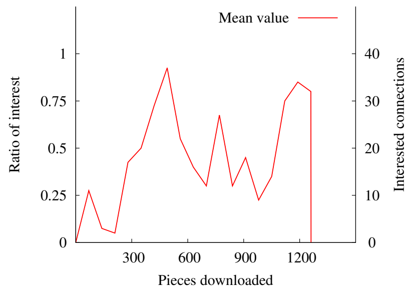


(a) *EnhancedBit*

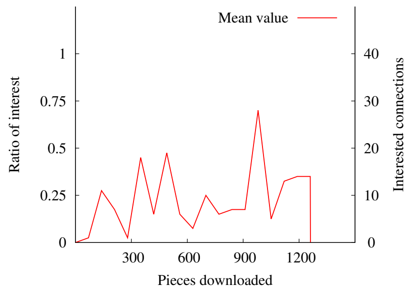


(b) Native *BitTorrent*

Figure 5: Ratios of interest of peers and interested connections under (a) our *EnhancedBit* and (b) the native *BitTorrent* protocol, as a function of downloaded pieces.



(a) *EnhancedBit*



(b) Native *BitTorrent*

Figure 6: Average ratio of interest of peers and interested connections under (a) our *EnhancedBit* and (b) the native *BitTorrent* protocol, as a function of downloaded pieces.

1) Figures 5(a) and 5(b) illustrate the *ratios of interest* and the number of *interested connections* maintained by peers, as a function of downloaded pieces. Figures 6(a) and 6(b) illustrate the average *ratio of interest* of peers and *interested connections*, as a function of downloaded pieces. From input files of Figures 6(a) and 6(b), we calculate that during the experiments the average *ratio of interest* was 0.45 and 0.30 per peer, under the *EnhancedBit* and the native *BitTorrent* protocol. We use equations (3), (4) from Section 5 to calculate p, q, and variance (σ^2_{RI}) of *ratio of interest*. We have:

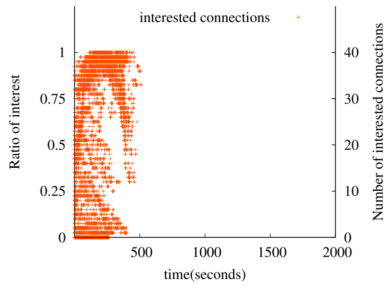
- *EnhancedBit*: $(p, q, \mu, \sigma^2) = (0.45, 0.55, 0.45, 0.24)$
- Native *BitTorrent*: $(p, q, \mu, \sigma^2) = (0.30, 0.70, 0.30, 0.21)$

As defined in the Bernoulli trial of Section 5, p is the probability of the event that a peer i needs a piece from a neighboring peer j. We observe that p is higher under the *EnhancedBit* protocol, which means that there is a more effective utilization of the rarest first algorithm (Section 3). Moreover, under *EnhancedBit* peers maintain a higher average *ratio of interest* and the values are spread with a higher variance near the respective mean value. To extend our view, in Table 5 we calculate the probability that a peer maintains more than, or equal to, x *interested connections*. The random variable X which expresses the number of *interested connections* maintained per peer follows the binomial distribution with $v = 40$. From equations (1), (2) we get $P(X \geq x) = 1 - F(x) = 1 - \sum_{k=1}^{x} \binom{40}{k} p^k (1-p)^{40-k}$, where $p = 0.45$ or 0.30. Table 5 indicates that under the *EnhancedBit* protocol, peers maintain

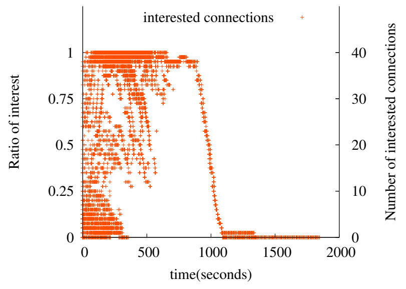
Interested Connections x	*EnhancedBit* $P(X \geq x)$	Native *BitTorrent* $P(X \geq x)$
5	1 or 100%	1 or 100%
10	0.99 or 99%	0.94 or 94%
15	0.96 or 96%	0.44 or 44%
20	0.56 or 56%	0.04 or 4%
25	0.07 or 7%	0.01 or 1%

Table 5: Cumulative distribution functions

a higher number of *interested connections*, than under the native counterpart. Furthermore, from the integral of the function that describes average *ratio of interest* we measure the aggregate benefit obtained under (a) our *EnhancedBit* and (b) the native *BitTorrent* protocol. It is trivial to say that in Figure 6(a) the area bellow the red line is larger, than that in figure 6(b). The above is consistent with proposition 1 where we analyze that $B_{EN\_BT} > B_{BT}$.



(a) *EnhancedBit*



(b) Native *BitTorrent*

Figure 7: Ratios of interest of peers and interested connections under (a) our *EnhancedBit* and (b) the native *BitTorrent* protocol, as a function of time.

2) Figures 7(a) and 7(b) illustrate the *ratios of interest* and number of *interested connections* maintained by peers over the duration of the experiment. In Figure 7(a) (*EnhancedBit*) the samples of *interested connections* are spread until the 500*th* second of the experiment when all peers finish downloading. On the other hand, in Figure 7(a) (native *BitTorrent*), *interested connections* are spread well past this point in time and approximate the 2000*th* second of the experiment as soon as all peers finish downloading. In the first case, all peers act as intermediaries (downloading **and** uploading data) 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, *EnhancedBit* 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, 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 additional clients. 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.

Experiment 2: Bootstrapping Period – Downloading time

Second, we examine the bootstrapping period and downloading completion time achieved by leechers, under the *EnhancedBit* and the native *BitTorrent* protocol. To this end, we set two swarms formed under flash-crowd conditions: 120 leechers join each swarm during the time period between 0 and 10 seconds of our experiment. In both configurations used –native and *EnhancedBit*– leechers remain idle before receiving their first *optimistic unchoke* interval. We refer to this initial time as the bootstrapping period.

In Section 5, we show that 100% of the *EnhancedBit* leechers and 63.7% of the native *BitTorrent* leechers should be unchoked within 30 seconds. Figure 8

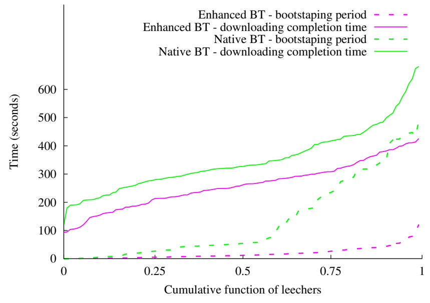


Figure 8: Bootstrapping periods and downloading completion times of peers under (a) our *EnhancedBit* and (b) the native *BitTorrent* protocol.

shows that under *EnhancedBit* approximately 90% of leechers receive their first optimistic unchoke interval within 30 seconds, whereas under the native *BitTorrent* approximately 55% of leechers receive an optimistic unchoke interval within 30 seconds. The deviation presented between theory and practice is reasonable due to the fact that theoretical analysis accounts for steady state swarms. However, during experimentation both swarms were not in steady state. In the native *BitTorrent* we discern a non-negligible number of leechers experiencing a lengthy bootstrapping period, while in the *EnhancedBit* this metric eases considerably. 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 8 shows that leechers that experience a lengthy bootstrapping period, also experience a dramatically increased downloading completion time. Under *EnhancedBit*, all leechers have completed downloading by the 350-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. Fresh and/or underutilized 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 both bootstrapping period and downloading completion time achieved by leechers.

Experiment 3: Unchokes – Interested messages – Uploaded Data

In Figure 9, we examine the correlation between unchokes received by leechers (top subplots), *interested* messages received by leechers (middle subplots), and the volume of data uploaded by leechers (bottom subplots). We compare against leechers of the *EnhancedBit* and the native *BitTorrent* protocol. Both swarms are formed from 120 leechers and 15 seeders. The former join the swarm at random times within 100 seconds. We order leechers according to their peer ID and the same order is kept for all subplots.

From Figure 9 –top subplots– we discern that in both settings used, leechers receive almost the same portion of unchokes (optimistic **and** regular). This is to be expected, since the timing settings used in the *EnhancedBit* client (Table 4) are inline with the directives of the native *BitTorrent* protocol [5]. Under the unbiased selection of the native *BitTorrent* protocol, optimistic unchoking slots are randomly distributed to peers. In contrast, under our *EnhancedBit* protocol, optimistic unchoking slots are allocated selectively to underutilized peers. The latter obtain data to disseminate and ultimately trigger the interest of additional peers. Figure 9 –middle subplots– confirms that *EnhancedBit* leechers exchange a drastically increased number of *interested* messages, compared to native *BitTorrent* leechers. Peer that are receiving many *interested* messages will ultimately contribute uploading capacity to the swarm. In contrast, peers that are receiving few *interested* messages will in all likelihood not act as uploaders and shall remain idle. As shown in Figure 9 –bottom subplots– *EnhancedBit* leechers disseminate a notably greater volume of data, than native *BitTorrent* counterparts. It is worth pointing out that in both settings used –native and

EnhancedBit– the portion of data each leecher uploads is proportional to the amount of *interested* messages the leecher under examination receives.

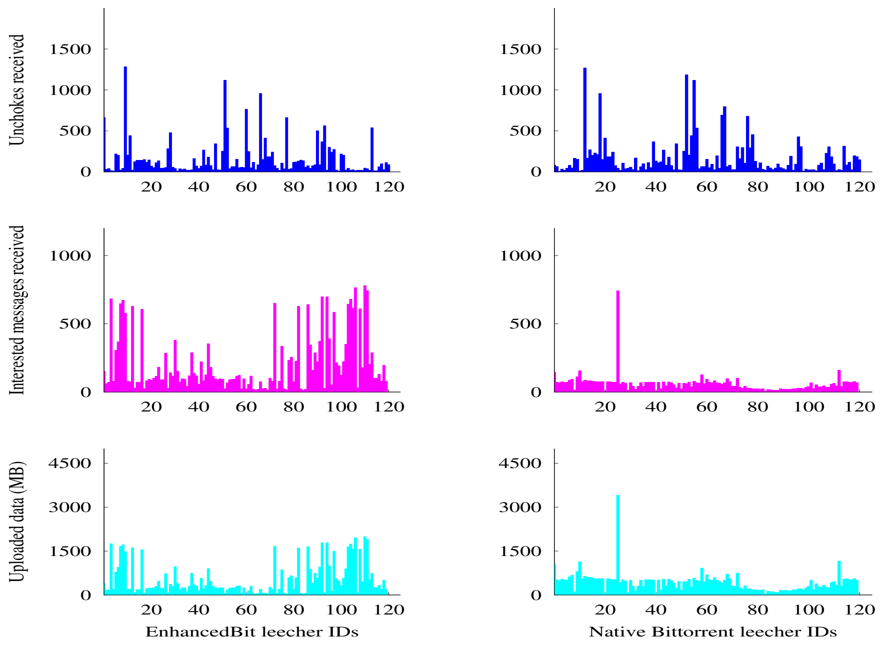
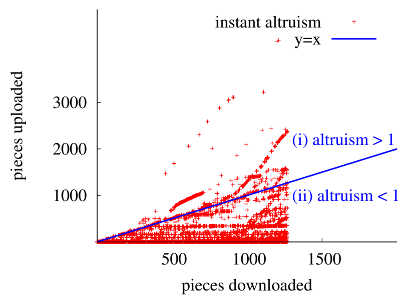


Figure 9: Correlation of unchokes, interested messages and uploaded data of leechers under (a) our *EnhancedBit* and (b) the native *BitTorrent* prtocol.

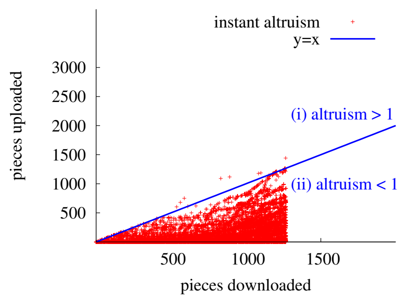
Experiment 4: Altruism of Leechers – Aggregate Uploading Contribution

In this experiment, we examine the *altruism* presented by leechers, and compare the aggregate uploading contribution of leechers against seeders. *Altruism* is defined as the ratio: *pieces uploaded/pieces downloaded*. Again, we compare leechers of *EnhancedBit* with leechers of the native *BitTorrent* protocol. The swarms are formed from 120 leechers and 15 seeders.

Figures 10(a) and 10(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. Although, in *EnhancedBit* (Figure 10(a)) there is a non-negligible number of peers clustered in area (i), there are only a handful of peers in the same area in the native *BitTorrent* (Figure 10(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 and provide the swarm with additional uploading capacity. To gain a global perspective, in Figure 11 we compare the aggregate uploading contribution of leechers and seeders under the *EnhancedBit* and the native *BitTorrent* protocol. Under



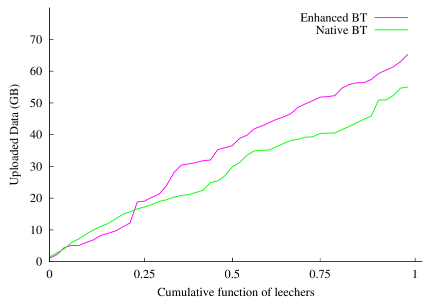
(a) *EnhancedBit*



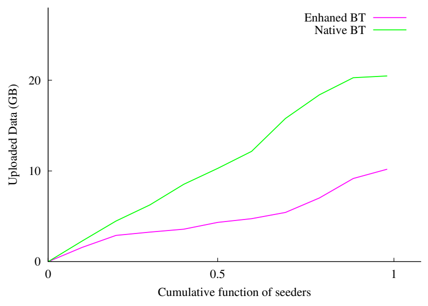
(b) native *BitTorrent*

Figure 10: Altruism presented by leechers under (a) the *EnhancedBit* and (b) the native *BitTorrent* protocol.

the native protocol, seeders upload $20GB$ of data and leechers upload $55GB$ of data. Under the *EnhancedBit* protocol, seeders upload $10GB$ and leechers upload $65GB$. These leechers decongest seeders and provide the swarm with additional uploading capacity of up to $10GB$. Our approach thus achieves an increase in the contribution of leechers, without involving any complex incentive policy. This is inline with our key objective to encourage underutilized peers to act as data intermediaries, rather than penalize them.



(a) Leechers



(b) Seeders

Figure 11: Aggregate Uploading Contribution of (a) leechers and (b) seeders; under our *EnhancedBit* (purple line) and the native *BitTorrent* protocol (green line).

Discussion: Free riders

Finally, we consider the case in which one or more free riders are trying to download data without uploading any. Under our *EnhancedBit* protocol, it is trivial to implement a "hacked" free rider, i.e., a strategically modified *EnhancedBit* client with an intent to trick a swarm of rational peers. This "hacked" free rider may send *have* messages indicating low *ratio of interest* to monopolize optimistic unchokes and complete downloading earlier than well-behaved peers, without uploading any data. To penalize "hacked" free riders under our *EnhancedBit* protocol, we can customize a simple token-based policy introduced in [22]. The authors of [22] suggested AntFarm: an efficient Content Delivery Network under 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 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. AntFarm's technique is applicable to our *EnhancedBit* protocol. "Hacked" free riders 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 *EnhancedBit* protocol is the one proposed in [27]. 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). "Treat-before-trick" (TBeT) is applicable to our *EnhancedBit* as well.

7. Conclusion

In this paper, we present the *EnhancedBit* protocol whose *unchoking policy* better harnesses underutilized peers that have few clients interested in downloading data from them. Our *EnhancedBit* protocol involves uploaders allocating *optimistic unchoking* slots to underutilized clients. This policy enables clients to obtain data and essentially act as content intermediaries, rather than remain idle. To measure performance gain under our approach, we have introduced an analytical model and compared our *EnhancedBit* with the native *BitTorrent* protocol. Experimentation with leecher and tracker prototypes shows that our approach achieves improved quality of inter-connections amongst peers compared with the native *BitTorrent* protocol. Under our *EnhancedBit* protocol, the number of directly-connected and interested-in-cooperation peers increases significantly. Moreover, our *EnhancedBit* protocol has the effect of creating altruistic leechers who act more as uploaders than downloaders. Therefore, a substantial portion of the leechers act as data intermediaries and 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 *EnhancedBit* achieves a shorter bootstrapping period and a shorter downloading time, *without* the use of complex incentive policies.

Acknowledgments: We would like to thank the anonymous reviewers for their fruitful feedback. This work was partially funded by the *iMarine EU-FP7* project and ERC Starting Grant # 279237. Part of this work appeared in the 18th International European Conference on Parallel and Distributed Computing [8].

References

[1] Bernoulli Trials and Binomial Distribution. `http://www.encyclopediaofmath.org/index.php/Bernoulli_trials`.

[2] Internet Study. `http://www.ipoque.com/en/resources/internet-studies/`.

[3] IsoHunt. `http://isohunt.com/`.

[4] Napster. `http://music.napster.com/`.

[5] The BitTorrent Protocol. `http://www.bittorrent.org/`.

[6] TorrentLeech. `http://torrentleech.org/`.

[7] Azureus(Vuze). `http://azureus.sourceforge.net/`, January 2011.

[8] V. Atlidakis, M. Roussopoulos, and A. Delis. Changing the Unchoking Policy for an Enhanced BitTorrent. In *Euro-Par*, pages 377–388, Rhodes, Greece, August 2012.

[9] B. Cohen. Incentives Build Robustness in BitTorrent. In *USENIX IPTPS*, Berkeley, CA, USA, February 2003.

[10] A.R. Bharambe, C. Herley, and V.N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *IEEE INFOCOM*, pages 1–12, Barcelona, Catalunya, Spain, April 2006.

[11] J.L. Chiang, Y. Tseng, and W.T. Chen. Interest-Intended Piece Selection in BitTorrent-like Peer-to-Peer File Sharing Systems. *Journal of Parallel and Distributed Computing*, 71(6):879–888, June 2011.

[12] Alix L.H. Chow, L. Golubchik, and V. Misra. BitTorrent: An Extensible Heterogeneous Model. In *IEEE INFOCOM*, pages 585–593, Rio De Janeiro, Brazil, April 2009.

[13] Y. Hu, L.N. Bhuyan, and M. Feng. Peer-to-Peer Indirect Reciprocity via Personal Currency. *Journal of Parallel and Distributed Computing*, 72(8):1045–1054, August 2012.

[14] J.R. Jiang, J.S. Chiou, and S.Y. Hu. Enhancing Neighborship Consistency for Peer-to-Peer Distributed Virtual Environments. In *IEEE ICDCS-Workshops*, pages 71–76, Toronto, ON, Canada, June 2007.

[15] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *ACM SIGCOMM-Workshops*, pages 116–121, Philadelphia, PA, USA, August 2005.

[16] R. Landa, D. Griffin, R.G. Clegg, E. Mykoniati, and M. Rio. A Sybil-proof Indirect Reciprocity Mechanism for Peer-to-Peer Networks. In *IEEE INFOCOM*, pages 343–351, Rio De Janeiro, Brazil, April 2009.

[17] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and Sharing Incentives in BitTorrent Systems. In *ACM SIGMETRICS*, pages 301–312, San Diego, CA, USA, June 2007.

[18] W.C. Liao, F. Papadopoulos, and K. Psounis. Performance Analysis of BitTorrent-like Systems with Heterogeneous Users. *Performance Evaluation*, 64(9-12):876–891, September 2007.

[19] Z. Ma and D. Qiu. A Novel Optimistic Unchoking Algorithm for BitTorrent. In *IEEE CCNC*, pages 1–4, Las Vegas, NV, USA, January 2009.

[20] D.S. Menasché, L. Massoulié, and D.F. Towsley. Reciprocity and Barter in Peer-to-Peer Systems. In *IEEE INFOCOM*, pages 1505–1513, San Diego, CA, USA, March 2010.

[21] M.Yang and Y. Yang. An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing. *IEEE Transactions on Computers*, 59(9):1158–1171, September 2010.

[22] R. Peterson and E.G. Sirer. AntFarm: Efficient Content Distribution with Managed Swarms. In *USENIX NSDI*, pages 107–122, Boston, MA, USA, April 2009.

[23] M. Piatek, T. Isdal, T.E. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *USENIX NSDI*, Cambridge, MA, USA, April 2007.

[24] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *ACM SIGCOMM*, pages 367–378, Portland, OR, USA, August 2004.

[25] R. Rahman, T. Vink, D. Hales, J.A. Pouwelse, and H.J. Sips. Design Space Analysis for Modeling Incentives in Distributed Systems. In *ACM SIGCOMM*, pages 182–193, Toronto, ON, Canada, August 2011.

[26] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang. TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client. In *IEEE INFOCOM*, pages 1523–1531, San Diego, CA, USA, March 2010.

[27] K. Shin, D.S. Reeves, and I. Rhee. Treat-before-Trick: Free-Riding Prevention for BitTorrent-like Peer-to-Peer Networks. In *IEEE IPDPS*, pages 1–12, Rome, Italy, May 2009.

[28] M. Sirivianos, J.H. Park, R. Chen, and X. Yang. Free-riding in BitTorrent Networks with the Large View Exploit. In *USENIX IPTPS*, Bellevue, WA, USA, February 2007.

[29] Y. Tian, D. Wu, and K.W. Ng. Modeling, Analysis and Improvement for BitTorrent-Like File Sharing Networks. In *IEEE INFOCOM*, pages 1–11, Barcelona, Catalunya, Spain, April 2006.

[30] 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.

[31] 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 *IEEE ICDCS-Workshops*, pages 29–35, Toronto, ON, Canada, June 2007.

[32] C. Zhang, P. Dhungel, D. Wu, Z. Liu, and K. W. Ross. BitTorrent Darknets. In *IEEE INFOCOM*, pages 1460–1468, San Diego, CA, USA, March 2010.