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 novel optimistic unchoking approach for the *BitTorrent* protocol whose key objective is to improve the quality of inter-connections amongst peers. In turn, this yields enhanced data distribution without penalizing underutilized and/or idle peers. The suggested 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 consequently, 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" as they 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 leading to improved peer inter-connection. Experimental results indicate that our approach significantly outperforms the existing optimistic unchoking policy.

Keywords: Peer-to-peer, Content Distribution, Unchoking.

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 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* [2] is the most popular *peer-to-peer* protocol, accounting for approximately 27-55% of all Internet traffic depending on geographical

C. Kaklamanis et al. (Eds.): Euro-Par 2012, LNCS 7484, pp. 377–388, 2012.

location, according to [1]. In the pre-*BitTorrent* era, Napster, Gnutella and FastTrack 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 [2] operates at 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 allocates downloading slots 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 [2]. 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* initiated 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. These include the number of peers acting as intermediates, decongestion in seeders, contribution of aggregate seeders and peers, 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 under-utilized 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.

Although prior related research has been carried out in a number of aspects including reciprocity mechanisms [6, 8], *tit-for-tat* schemas to discourage free riding [14], and incentives policies in [12, 7], 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 upload bandwidth offered by peers. 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 2 discusses the key features of our proposed enhanced *BitTorrent* scheme and Section 3 presents our main experimental results. Section 4 outlines related work while concluding remarks are found in Section 5.

2 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 introduce and analyze the ratio of interest, and the algorithms used for our unchoking policy. Finally, we give an overview of our enhanced *BitTorrent* system.

2.1 Enhanced *BitTorrent* Messages

The messages of the native *BitTorrent* protocol can be categorized into: *swarm-oriented, state-oriented* and *data-oriented* messages. To implement our enhanced *BitTorrent* 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 2.2) of the sender of the *have* message and helps us implement our enhanced unchoking policy.

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.

Table 1. Swarm-oriented Messages

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.

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 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. 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.

Finally, Table 3 summarizes the *data-oriented* messages that are sent between *unchoked* peers (i.e., peers that are exchanging data).

2.2 Peer Unchoking - Ratio of Interest

We define the ratio of interest RI_p of a peer p to be $RI_p = \frac{int_p}{n_p}$, where int_p is the number of interested connections p maintains, from a total of n_p initiated connections. 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 initiated connections marked

Table 2. State-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 an integer identifying received piece, and a float corresponding to *ratio of interest* of 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 3. Data-oriented Messages

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.

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. To prevent peers from remaining idle, 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 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 3 where we compare the unchoking policies of our enhanced *BitTorrent* and the native *BitTorrent* protocol.

2.3 Algorithms

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 [2]. 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 *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. 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 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 $\leftarrow \{p : p \in Top3\ Uploaders\}$
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**

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. 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: $Downloaders, RI_{p \in Downloaders}$
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**

2.4 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 *peer set* of, typically, 50 peers to whom to connect. Afterwards, the fresh peer establishes *TCP* connections with peers in its peer set. Each peer is multi-threaded and asynchronously downloads/uploads data from/to multiple counterparts, without exceeding a threshold of 40 *initiated* connections. 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.

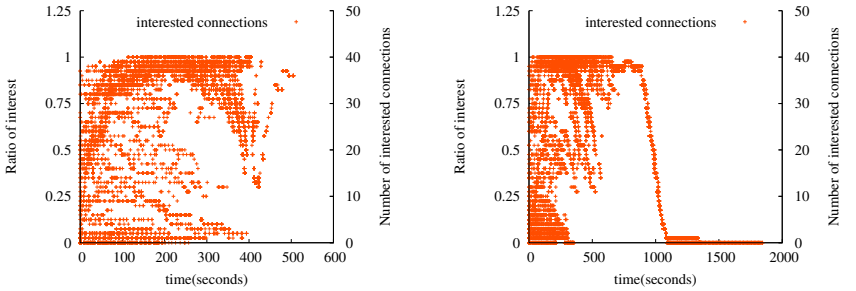
3 Evaluation

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: **a)** 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*, **b)** examine pieces uploaded from leechers and seeders to evaluate the decongestion of seeders achieved by our enhanced *BitTorrent*, and **c)** ascertain the degree of altruism attained by leechers in our enhanced *BitTorrent*. During experimentation we used an $700MB$ test file, $512KB$ pieces were shared among peers and each peer maintained 40 initiated connections. In steady state a swarm of as many as 150 peers was formed. 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.

Ratio of Interest

In this section, we examine the *ratio of interest* of peers, as defined in section 2.2. 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 Figs. 1(a) and 1(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



(a) Enhanced *BitTorrent* (b) native *BitTorrent*

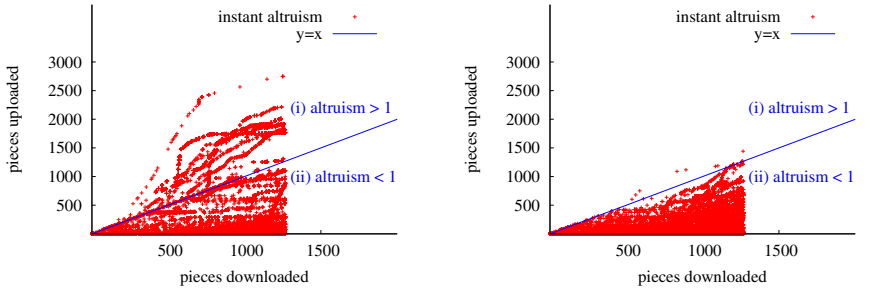
Fig. 1. Ratios of interest of peers and *Interested* connections under (a) our enhanced and (b) the native *BitTorrent* protocol. Initiated connections maintained per-peer are fixed at 40 and the ratio of interest is $RI \leq 1$ for both cases. The average ratio of interest is at 0.30 and 0.22 in (a)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

and 15 seeders; 90% of peers join a swarm within 100 seconds. In Fig. 1(a), which corresponds to the enhanced *BitTorrent* protocol, the average ratio of interest is 0.30 per peer, while in Fig. 1(b), which corresponds to the native *BitTorrent* protocol, the average ratio of interest is 0.22 per peer. Moreover, before 500 seconds in Fig. 1(a), there is a higher coverage of *interested* connections than that of Fig. 1(b). 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 native counterpart. An improved interconnection 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.

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 2(a) and 2(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) $altruism \geq 1$ and (ii) $altruism \leq 1$. 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* (Fig. 2(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* (Fig. 2(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



(a) enhanced *BitTorrent* (b) native *BitTorrent*

Fig. 2. 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 non-altruistic behavior ($altruism < 1$).

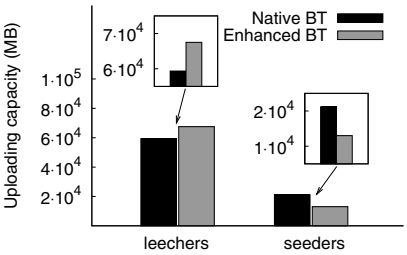


Fig. 3. Aggregate uploading contribution of leechers and seeders under the enhanced and the native *BitTorrent* protocol. Under the enhanced *BitTorrent* protocol, leechers upload 68 GB of data and under the native *BitTorrent* protocol leechers upload 58GB of data.

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 Fig. 3 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.

4 Related Work

A number of techniques have been suggested to improve the performance of the native *BitTorrent* [3] protocol including bartering-based approaches among peers, and incentive-based policies. In [6, 8], 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 [12], 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 [7], 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 [14, 16, 11]. 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 locating 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 [5]. To compliment neighborship, in our enhanced *BitTorrent* protocol we define *ratio of interest* (Section 2.2) and use this metric to decide which peer should be selected as planned optimistic unchoked. In [15], 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.

There have also been proposals for new models that essentially suggest *BitTorrent*-like protocols [4, 13, 10, 11, 9]. 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 and decongests seeders.

5 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 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.

Acknowledgments. We would like to thank Y. Mimiyannis, Y. Kamonas, C. Christou and A. Sevastidou for their help during our experimentation as well as the anonymous reviewers for their fruitful feedback. This work was partially funded by the *iMarine EU-FP6* project.

References

[1] Internet Study. http://www.ipoque.com/en/resources/internet-studies
[2] The Bittorrent Protocol. http://www.bittorrent.org/beps
[3] B. Cohen: Incentives Build Robustness in BitTorrent. In: IPTPS. Berkeley, CA (February 2003)
[4] Chow, A.L.H., Golubchik, L., Misra, V.: BitTorrent: An Extensible Heterogeneous Model. In: INFOCOM. pp. 585–593. Rio De Janeiro, Brazil (April 2009)
[5] Jiang, J., Chiou, J., Hu, S.: Enhancing Neighborship Consistency for Peer-to-Peer Distributed Virtual Environments. In: IEEE–ICDCS Workshops. Toronto, Canada (June 2007)
[6] Landa, R., Griffin, D., Clegg, R., Mykoniati, E., Rio, M.: A Sybilproof Indirect Reciprocity Mechanism for Peer-to-Peer Networks. In: INFOCOM. pp. 343–351. Rio De Janeiro, Brazil (April 2009)
[7] Legout, A., Liogkas, N., Kohler, E., Zhang, L.: Clustering and Sharing Incentives in BitTorrent Systems. In: SIGMETRICS. pp. 301–312. San Diego, CA (June 2007)
[8] Menasché, D., Massoulié, L., Towsley, D.: Reciprocity and Barter in Peer-to-Peer Systems. In: INFOCOM. San Diego, CA (March 2010)
[9] Meulpolder, M., Epema, D.H., Sips, H.: Replication in bandwidth-symmetric BitTorrent Networks. In: 22nd IEEE Int. Parallel and Distributed Processing Symposium (IPDPS'08). pp. 1–8. Miami, FL (April 2008)
[10] M.Y., Yang, Y.: An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing. IEEE Transactions on Computers 59(9), 1158–1171 (September 2010)
[11] Peterson, R., Sirer, E.: AntFarm: Efficient Content Distribution with Managed Swarms. In: NSDI. pp. 107–122. Boston, MA (April 2009)
[12] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., Venkataramani, A.: Do Incentives Build Robustness in BitTorrent? In: 4th USENIX Symposium on Networked Systems Design & Implementation. Cambridge, MA (April 2007)
[13] Ren, S., Tan, E., Luo, T., Chen, S., Guo, L., Zhang, X.: TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client. In: INFOCOM. pp. 1523–1531. San Diego, CA (March 2010)
[14] Shin, K., Reeves, D., Rhee, I.: 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). pp. 1–12. Rome, Italy (May 2009)
[15] Vassilakis, D.K., Vassalos, V.: An Analysis of Peer-to-peer Networks with Altruistic Peers. Peer-to-Peer Networking and Applications 2(2), 109–127 (June 2009)
[16] Yang, M., Feng, Q., Dai, Y., Zhang, Z.: A Multi-Dimensional Reputation System Combined with Trust and Incentive Mechanisms in P2P File Sharing Systems. In: IEEE–ICDCS Workshops. Toronto, Canada (June 2007)