A Survey of BitTorrent Performance

Raymond Lei Xia and Jogesh K. Muppala, *Senior Member, IEEE*

***Abstract*—Since its inception, BitTorrent has proved to be the most popular approach for sharing large files using the peer-to-peer paradigm. BitTorrent introduced several innovative mechanisms such as tit-for-tat (TFT) and rarest first to enable efficient distribution of files among the participating peers. Several studies examining the performance of BitTorrent and its mechanisms have been published in the literature. In this paper, we present a survey of performance studies of BitTorrent from 2003 to 2008. We categorize these studies based on the techniques used, the mechanisms studied and the resulting observations about BitTorrent performance. Many of the performance studies also suggested modifications to BitTorrent's mechanisms to further improve its performance. We also present a survey of the suggested improvements and categorize them into different groups.**

***Index Terms*—BitTorrent, Large file downloading, Peer-to-peer Networks.**

I. Introduction

THE PEER-TO-PEER (P2P) paradigm has proved to be a very effective approach for designing scalable and robust networking applications. This approach overcomes the scalability limitations of the traditional client/server approach. Large scale file-sharing and file-distribution applications were the first to exploit the new paradigm. BitTorrent [1], one of the most popular peer-to-peer (P2P) file distribution applications, was introduced by Bram Cohen in 2001 [2]. BitTorrent was quickly embraced by several content providers such as Lindows, Blizzard and most Linux distributions as a scalable way of delivering content, decreasing the load on congested servers, minimizing the distribution cost and reducing the downloading time for users. BitTorrent is a second generation P2P file sharing system [3], which focuses on organizing the peers that are interested in downloading the same file into an overlay network, called a *Torrent*. Interestingly, as more users join a Torrent, the downloading rate that is achieved by all the peers increases. BitTorrent is noted for introducing several innovative mechanisms including tit-for-tat (TFT) and rarest first (RF) that it uses for cooperative file distribution among the participating peers.

Any file distribution application implemented using the P2P paradigm requires two different functions to be supported: (1) a search function that enables peers to locate the content that they are interested in among the participating peers, and (2) a downloading function that enables the peer to download the content once it is located. Traditional P2P file sharing systems such as Gnutella [4] and KaZaa [5] focus on quickly locating the peers that possess a given file. Once located, the file is directly downloaded from one of the peers having the file. BitTorrent, on the other hand relies on cooperation among the peers for file distribution using swarming [2]. BitTorrent differs from the conventional P2P file sharing systems in three aspects. First, BitTorrent application does not provide a search function. Instead, peers are expected to peruse central-directory based search facilities provided by several supporting websites. Second, it provides a file-level sharing policy instead of a directory-level sharing policy. Finally, it employs a bartering mechanism such as TFT among peers [6], such that unless a peer contributes to the ongoing downloading of file pieces, it will not be able to download the file.

The widespread popularity of BitTorrent has attracted the attention of several researchers who conducted various performance studies in order to understand the behavior of the BitTorrent protocol, its mechanisms and the overall application performance. These studies affirmed the importance of TFT and rarest first mechanisms for BitTorrent's success. At the same time, several of these studies identified the impact of BitTorrent's phenomenal success on the traffic in computer networks. For example, one of the significant problems is the increase in file downloading traffic in the Internet. Reports show that BitTorrent has become the largest source of P2P file sharing traffic in the Internet: BitTorrent traffic accounted for 18% of broadband traffic in 2006 [7]. Karagiannis et al. [8] also report that BitTorrent occupies about 13-15% of the access link bandwidth at a residential university. Furthermore, BitTorrent presents significant traffic-engineering challenges for Internet service providers (ISPs). Current implementations of BitTorrent ignore the underlying Internet topology or ISP link costs and result in a large amount of cross-ISP traffic [9]. These issues show that BitTorrent protocol has further scope for improvement in its core mechanisms. For example, one of the most criticized of BitTorrent's mechanisms is the random neighbor selection strategy, which prolongs the file downloading time and results in inefficient usage of network resources. Some researchers also suggested BitTorrent-like protocols [10], [11], [12], which modify the BitTorrent mechanisms to further improve performance.

In this paper, we present a comprehensive survey of all the performance studies on BitTorrent. We summarize the salient findings of these performance studies. We then present a survey of all the improvements that have been suggested for BitTorrent and its mechanisms by various researchers. We categorize these improvements into various groups so as to enable us to understand the interplay among the suggested

Manuscript received 17 June 2008; revised 14 December 2008. The work described in this paper has been supported by HK RGC under RGC-Competitive Earmarked Research Grant HKUST 617907

R. L. Xia is with the Dept. of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: xialei@cse.ust.hk).

J. K. Muppala is with the Dept. of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: muppala@cse.ust.hk).

Digital Object Identifier 10.1109/SURV.2010.021110.00036

improvements. Surveys of P2P networks in general, like [13] and [14], application-layer multicast protocols [15], P2P security issues [16], and P2P based resource discovery [17] are already available in the literature. The difference between these surveys and our paper is that we focus specifically on BitTorrent, which is one of the most popular P2P applications.

The remainder of this paper is organized as follows. In section II, we present an overview of BitTorrent and its mechanisms. In Section III, we present a survey of the studies on BitTorrent performance. Section IV reviews several of the improvements suggested for BitTorrent mechanisms to further improve its performance. Then we compare the suggested improvements and categorize them into several groups. Finally, we conclude this paper in section V.

II. Overview of BitTorrent

A. What is BitTorrent

As stated earlier BitTorrent (BT) is one of the most popular applications for distributing large size files. The application is implemented as a hybrid P2P system, with most of the interaction directly among the peers, but requiring occasional interaction with a server for locating peers. The BitTorrent protocol requires peers to organize themselves into an overlay network, with connections among the peers, for each file being distributed. This overlay network is called a *Torrent*. Each file being distributed by BitTorrent requires the establishment of separate torrent. Besides the peers, a tracker and a web server play an important part in file distribution using BitTorrent.

The *tracker* is a special infrastructure node which stores meta-information about the peers that are currently active within a torrent. Peers interact with the tracker using a simple protocol layered on top of HTTP in which a peer sends information about the file it is downloading and the port number [2] to the tracker. The tracker does not participate in the actual distribution of the file, but only serves the purpose of enabling peers to find each other.

The peers that are part of a torrent can be classified into two types: a *seed* and a *leecher*. A seed is a client that has a complete copy of the file and remains in the torrent to serve other peers. For a torrent to get started, we need at least one *initial* seed that provides the entire content for download. A leecher is a client that is still downloading the file.

When a file is to be made available for distribution through a torrent, then a special file with a .torrent extension is made available on a web server. The .torrent file contains information about the file including its length, name, hashing information, and the url of a tracker [2]. A peer that wants to download a file first obtains the corresponding .torrent file from the web server. Then, the peer contacts the tracker and requests a list of IP/port pairs of other peers that are already participating in the torrent. When the tracker receives a request, it will send the requesting peer a list of potential neighboring peers, usually 50, that are selected randomly from the set of all active peers within the torrent. After obtaining the list, the peer contacts around 20–40 peers from the list to add them as its *neighbors*. This set of neighbors forms the *peer set* for the joining peer. If the number of neighbors connected to a peer falls below 20, the peer will again contact the tracker and obtain a new list. The data flow between two connected peers is bidirectional over the connection.

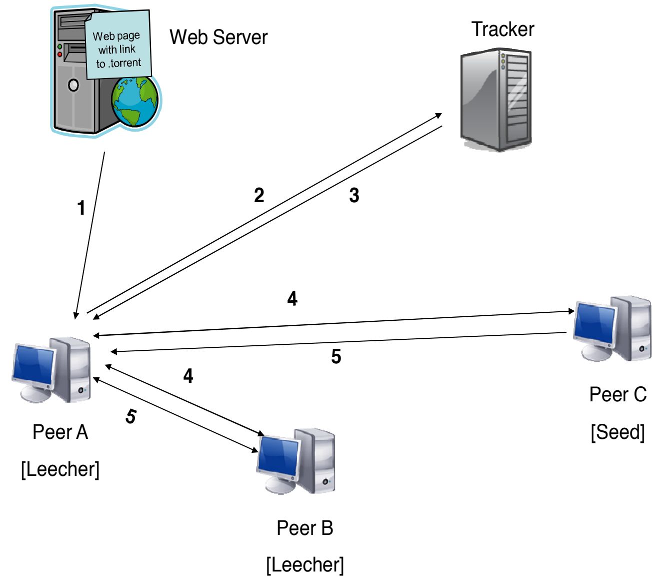

Fig. 1. BitTorrent file sharing process

The steps in file sharing using BitTorrent are shown in Fig. 1. There are a total of five steps:

1) When peer A wants to join a torrent, it first downloads the corresponding .torrent file from a web server.
2) Then, peer A contacts the tracker and asks for a list of peers that are already participating in the torrent.
3) Next, the tracker sends a list of about 50 peers which are participating in the torrent.
4) Then, peer A selects 35 peers from the list as its neighbors, and establishes connections with them.
5) After the connections are established, peer A can exchange file pieces with the neighbours.

The file exchange among peers uses a swarming technique [18], in which the file is broken into fixed size pieces (typically 256 KB each), and peers exchange pieces with each another. When a piece is completely downloaded, its SHA1 hash is computed and compared with the value in the metafile (.torrent). If the values match, the peer will announce the availability of the complete piece to its neighbours.

BitTorrent uses pipelining in order to keep its TCP connections operating at full capacity. To facilitate this, a piece is divided into sub-pieces, typically of 16 KB in size, which are called *blocks* or *chunks*. Requests for sub-pieces are kept pending in a pipeline, typically five in number. Each sub-piece arrival will trigger a request for another sub-piece. This way the connection is kept busy most of the time.

A peer downloads pieces not only from the seed(s), but also from other peers, thereby substantially reducing the load on the seed(s). A peer usually can serve four peers simultaneously, and it chooses the best four peers to *unchoke* and chokes other requesting peers. *Choking* is a temporary refusal to upload, but the connections are not closed.

B. Mechanisms of BitTorrent

The mechanisms employed by BitTorrent mainly consist of peer and piece selection strategies. A good peer selection

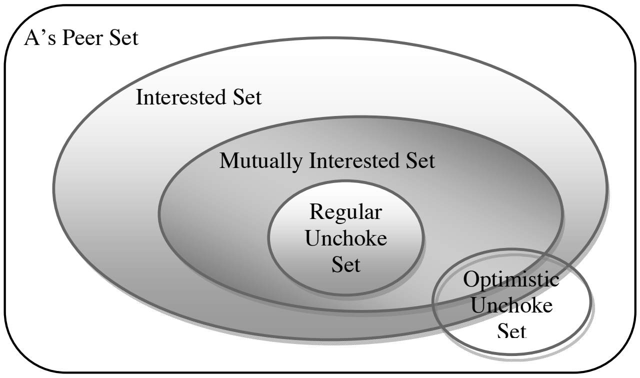

Fig. 2. Peer Sets

strategy should maximize the service capacity of the system, and an efficient piece selection strategy should guarantee that each peer can find pieces that are interesting to it from its neighbors. A detailed description of these mechanisms is given in [2]. Here we briefly summarize these mechanisms for completeness so that further discussions in the subsequent sections can be placed in proper context.

As already mentioned in the previous section, the peer set of a peer A is the set of all the neighbors of A. Peer A will consider a peer B belonging to its peer set as an *interested* peer, if Peer B has pieces that A does not possess. We can define an *Interested Set* of Peer A as the set of interested peers, which is a subset of its peer set. A peer can continue its download only if its interested set is non-null. Ideally Peer A should use appropriate mechanisms to maximize the size of its interested set. Doing so increases the ability of Peer A to download file pieces from the maximum number of other peers. Similarly, Peer A considers peer B as a *mutually interested* peer if and only if peer B is in peer A's interested set and peer A is also an interested peer for Peer B, i.e., Peer A has pieces that Peer B does not possess. It is beneficial to maximize the size of the mutually interested set because this increases the chances for trading pieces with the neighbors, thus effectively increasing the downloading rate.

A peer A considers a peer B as a member of its *regular unchoke* set if peer B is unchoked by Peer A using the TFT mechanism. Peer A considers peer B as a member of its *optimistic unchoke* set if peer B is unchoked by Peer A due to the optimistic unchoking mechanism. Details of these mechanisms are given later in this section. We illustrate these sets and their relationships further in Fig. 2. These sets play an important role in determining the downloading experience of any peer.

1) Peer Selection Strategy: The peer selection strategy uses four main mechanisms: tit-for-tat (TFT), optimistic unchoking (OU), anti-snubbing, and upload only. Summed up together, they form the basis for the *choking algorithm* used by a peer. The major aim of these mechanisms is to improve the downloading experience of those peers that contribute to the file exchange by uploading pieces to others, and punish *free-riders*, i.e., those peers that download pieces from others without uploading any pieces.

- Tit-for-tat: The tit-for-tat strategy is used by a peer to preferentially upload to its neighbors who provide it the best downloading rates. At any given time, a peer exchanges file pieces with a fixed number of neighboring peers, usually at most four. A leecher preferentially uploads to the best three neighbors who provide it the best downloading rate and chokes others. Every 10 seconds, the leecher reevaluates the downloading rate from all peers who are sending data to it. If another peer offers better downloading rate, the leecher will choke the peer with the smallest downloading rate among the current peers it is serving, and unchoke the better peer. This mechanism is very important to encourage contributors and punish free-riders, thus preventing leechers from downloading without contributing anything.
- Optimistic Unchoking: Besides the TFT mechanism, BitTorrent incorporates an optimistic unchoking policy. Every 30 seconds, a peer unchokes a neighboring peer randomly regardless of its uploading rate. The benefit of Optimistic Unchoking is that it enables a peer to discover better neighboring peers to exchange file pieces with since other neighbors may have higher capacity for uploading than the currently unchoked peers. If a peer uses only the TFT mechanism, there will be no opportunity for discovering other peers that can provide higher uploading rate. In addition, this strategy is especially useful for newly joined peers to get started.
- Anti-snubbing: A peer may be choked by the peers it was formerly downloading from, thereby getting poor downloading rates. To address this problem, when a peer notices that some time has elapsed without getting a single piece from a neighboring peer, the leecher assumes it is 'snubbed' by that peer and does not upload to it any further through regular unchoke.
- Upload Only: Once a peer finishes downloading the entire file, it becomes a seed. Since seeds have nothing to download, they cannot select peers based on downloading rates. Instead, the seeds prefer to upload to peers with better uploading rates [2]

2) Piece Selection Strategy: The second major strategy that plays an important role in the success of BitTorrent is the piece selection strategy. When a peer wants to download pieces from its neighbors, it adopts several piece selection strategies, which includes the following four mechanisms:

- Strict Priority: In BitTorrent, peers concentrate on downloading a whole piece before requesting another piece. Thus if a sub-piece is requested, then subsequent sub-pieces of the same piece will be requested preferentially in order to complete the download of the whole piece as soon as possible, because only complete pieces can be traded with others.
- Rarest First: Peers often prefer to download pieces which are the rarest among their neighbors [2]. This strategy is called the rarest first algorithm which works as follows. Each peer maintains a list of the pieces that each of its neighbors possess. This list is updated each time a copy of a piece becomes available from its neighbors. It then constructs a rarest-pieces set which is the list of those

pieces that have the least number of copies among its neighbors. It then selects the piece to download that is rarest among its neighbors. This strategy increases the chance of a peer trading with its neighbors as it has pieces that others want. This also benefits the seeds by reducing their server burden, since they need to serve out less copies of the same piece. Furthermore, the departure of a seed will not leave the remaining downloaders without the opportunity to exchange file pieces, because this strategy ensures that all the pieces are quickly distributed to at least some of the leechers [2].
- Random First Piece: There is an exception to the rarest first policy when a peer first joins a torrent. Since the peer does not have any pieces, so it would like to download a whole piece quickly so that it can get ready to reciprocate with the TFT algorithm. In this case, the new peer will not seek the rarest piece. Instead, it will download the first piece randomly in order to make sure it can get a whole piece as fast as possible.
- Endgame Mode: This mode is adopted by a peer towards the end of downloading the file. If a piece is requested from a peer that has a slow transfer rate, the downloading time will be prolonged. To address this problem, a peer requests all of its neighbors for blocks it has not received. Once a block is obtained, the peer cancels the request for that block from its neighbors so as to decrease bandwidth wastage due to redundant downloads.

C. BitTorrent-like Systems

Some researchers proposed BitTorrent-like systems such as Slurpie [12], Fox [11], and Avalanche [10]. They have a lot of similarities in their operations with BitTorrent. The common goal of these approaches is to improve the file sharing performance.

Slurpie [12] is designed with the goal of reducing the clients' downloading time and making the client sets scalable. In Slurpie, when a node wants to download a file, it first contacts a server called a topology server. The topology server sends the node a list of other nodes which are downloading the same file. The node then contacts the other nodes in the list to form a mesh. Similarly, the file is divided into fixed size blocks, and is exchanged through the mesh. The authors claim that Slurpie has advantages in adapting to varying bandwidth and achieves better scaling with the number of neighbors compared to BitTorrent. Slurpie employs an effective downloading bandwidth estimation technique described in [12] which reports three different states of bandwidth for each node: underutilized, at-capacity, and throttle-back. Each node in Slurpie maintains a target number of neighbors, and this number is continually updated according to the available bandwidth. The bandwidth estimation algorithm runs every second, and if the node's state is underutilized and the node needs more blocks, a new connection will be opened. By using these strategies, Slurpie can improve the system performance as the size of the network increases.

Levin et al. [11] proposed a Fair, Optimal eXchange protocol, called *FOX* to achieve a fair file swarming. FOX provides not only an effective application of the TFT mechanism, but also theoretically optimal downloading time for peers. In FOX, the authors assumed that all peers are greedy. In the overlay structure of FOX, there are N clients downloading a file from one server. Suppose R_i is the peer connected to the server, and T_i is the subtree rooted at R_i, and L_i is the set of leaves of T_i. To get a quick downloading time, FOX fills the available outgoing link at the server, in which each R_i requests a different block from the server, and each leaf only sends blocks to leaves which reciprocate the gesture. Peers in FOX have two properties. The first property is the give-and-take symmetry, which means each of peer's outgoing edges has a corresponding incoming edge, instead of allowing peers to arbitrarily match up. The second property is that the connections are stable. Based on these two properties, peers' misbehavior, such as receiving but not sending, can be detected. In addition, FOX can deal with the last-block collapse, in which the finished nodes quit the system leaving the remaining nodes without any help. Fox uses a cryptographic method, in which the final blocks are encrypted by the internal nodes before sending to the leaves. Then, the leaves use their own keys to encrypt the encrypted blocks. So both nodes and leaves need others' keys before decrypting the encrypted blocks and completion of download. By using this approach, both internal nodes and leaves hold information that the others need to complete their download. So FOX can make all nodes complete their download at the same time and avoid the last-block collapse.

In P2P systems, users often decide which block to download without considering whether the block has already been sent out by the server at least once or not. This causes suboptimal distribution decisions. Gkantsidis et al. [10] proposed *Avalanche* using networking coding technique. Like BitTorrent, nodes join the system by contacting a centralized server which provides a random subset of other users. Nodes also employ rarest first algorithm to decide which block to transfer. In Avalanche, there are two incentive mechanisms. The first one is giving priority to exchange over free uploading to other nodes. The second one is that nodes do not upload to users unless they have also received sufficient content in return, which is similar to the TFT policy. The principle of network coding is to allow nodes to encode packets. When a client wants to send a packet to others, it will make a linear combination of all the available information and then send it out. At the receiver's side, the original information will be reconstructed after receiving enough linear combination of packets. With this method, the block propagation becomes more efficient and the system is more robust under extreme situations like the sudden departure of nodes.

The characteristics of the above BitTorrent-like systems are summarized in Table I. As can be seen from the table, the above three systems not only share some mechanisms in common with BitTorrent, but also have their own characteristics. Slurpie is very close to BitTorrent in its operation. A major difference is that it provides bandwidth estimation technique which enables it to scale well as the group size increases. FOX proposes a fair protocol to provide an optimal downloading time. While Avalanche is very similar to BitTorrent, it uses network coding technique to make the scheduling of the content propagation easier.

TABLE I
BITTORRENT-LIKE SYSTEMS

BitTorrent-like system	Special Characteristics	Common characteristics with BT	References
Slurpie	Bandwidth estimation technique. The number of neighbors is updated according to the available bandwidth	Central server. Mesh formed among peers.	[12]
FOX	Peer connections are based on give-and-take symmetry. The connections are stable.	TFT policy	[11]
Avalanche	Each piece of the shared file is encoded by the nodes using network coding.	Central server. Rarest first policy. Incentive mechanism.	[10]

III. PERFORMANCE OF BITTORRENT

The popularity of BitTorrent has spurred a large number of studies of its performance. Some measure its behavior on real torrents; others use the simulation to observe its performance while others use mathematical analysis-based models to evaluate its performance. In this section, we first organize the papers based on the performance evaluation method used. The main purpose of organizing this section according to the performance evaluation methods used is to group together the papers using similar approaches to evaluation. We do notice that each evaluation technique lends itself to examining certain aspects of BitTorrent performance better than the other techniques. We highlight the advantages and shortcomings of each approach in the respective section. In the final subsection we concentrate more on reviewing the techniques from the perspective of the issues addressed and the different aspects of performance studied.

A. Measurement

Several measurement-based studies of BitTorrent performance have been presented in the literature. Most of the measurement-based experiments lasted for several months. Generally, the following five methods were used: (a) analyzing the tracker logs obtained from the trackers, (b) using scripts to gather information from torrent websites and directly from peers, (c) analyzing packet traces collected at Internet access link, (d) joining an ongoing torrent with a modified client specially instrumented to collect event logs, and (e) conducting experiments on network testbeds like PlanetLab or user-constructed networks of PCs. Observing the tracker log can enable us to get the information on the global view of BitTorrent performance; whereas using a modified client enable us to observe the individual behavior of peers.

Izal et al. [18] evaluated the performance of BitTorrent mechanisms based on five months of measurement data collected from BitTorrent peers. This was one of the first comprehensive evaluation of BitTorrent using measurement. First, they used the tracker log of Linux RedHat 9 distribution for their evaluation. They found that:

- BitTorrent clients are altruistic.
- Seeds contribute more than twice the data uploaded by leechers.
- The proportion of seeds is higher than 20%.

Then, they used an instrumented client to join the torrent to observe the individual behavior of a peer. They observed that:

- After obtaining the first few pieces, peers begin trading pieces soon, which indicates that the rarest first policy works well.
- The uploading and downloading rates are correlated, which means that the TFT policy works well.

We note that the perspective presented in this paper is based on the detailed analysis of a single torrent, which may be its limitation in that the perspective is limited. However the observations are still relevant and corroborated by other studies reviewed in this section later.

Pouwelse et al. [6] presented a measurement based study of BitTorrent's availability, integrity, flash crowd handling, and download performance based on measurement data obtained from June 2003 to March 2004. They also measured Suprnova, one of the most popular websites for BitTorrent clients to find files. Their experiment consisted of two parts. They monitored the global BitTorrent components using three scripts: a Mirror script for measuring the Suprnova mirrors' availability and response time, a HTML script for gathering and parsing Suprnova mirrors' HTML pages and downloading new files with the extension of .torrent, and a Tracker script used to parse the .torrent files for new trackers and check the trackers' status. They also observed the actual peers in the system using scripts: the Hunt script was used to select a file to follow and initiate a measurement of all the peers who are downloading that file, the Getpeer script was used to obtain the IP addresses of peers downloading the file from the tracker, the Peering script for measuring the download progress and uptime of peers in parallel. They found that:

- The number of users is related with the number of seeds in BitTorrent.
- Most users often stay in the system to be seeds for several hours after finishing their downloading.
- BitTorrent can deal with flashcrowds efficiently.
- Fewer number of seeds does not mean that the lifetime of the file is short. Even files with only one seed can still have a long content lifetime.

The authors concentrated mainly on the user behavior and the content distributed by BitTorrent. This gives a good macro view of BitTorrent behavior. However this measurement does not directly shed light on the core mechanisms in BitTorrent and their effect on its performance.

Legout et al. [22] conducted a detailed measurement study aimed at understanding the behavior of the rarest first algorithm and the choking algorithm on real torrents. They instrumented a BitTorrent client and observed its performance in several torrents. They observed the log of messages sent or received, the log of each state change, the log of the rate estimation, and the log of important events such as end game mode. Their results showed that (1) the rarest first algorithm prevents the reappearance of rare pieces and avoids

TABLE II
SUMMARY OF MEASUREMENT-BASED STUDIES OF BITTORRENT PERFORMANCE

Measurement Methods	Results	References
Observing the tracker log	BitTorrent clients are altruistic. The rarest first policy works well. The TFT policy works well	[18]
	BitTorrent can support large files. The incentive mechanisms are effective	[19]
	TFT is imperfect at preventing free-riding	[20]
	The distribution of performance among peers is roughly uniform. TFT is the main factor that affects the performance	[21]
Using Mirror script, HTML script, Tracker script, Hunt script, Getpeer script, and Peering script	BitTorrent can deal with flash crowds effectively	[6]
Analyzing packet traces collected at Internet access link of a residential university network, and Tracker Logs	BitTorrent is locality-unaware, causing the same content to be downloaded into the same locality (defined around an ISP) multiple times. Significant overlap in lifetimes of peers within the same locality. Incentives to peers to stay after download completes to serve others has significant impact on performance	[8]
Using a client on real torrent	The rarest first algorithm prevents the reappearance of rare pieces and avoids the last pieces problem. The choke algorithm is fair and robust.	[22]
	Peers behind a firewall have low bandwidth usage. Peers behind a firewall often disconnect early	[23]
Using a modified version of BT on PlanetLab	The choking algorithm can cluster peers with similar bandwidth, reward contributing peers, and make peers have a high upload utilization	[24]
	The initial stage is not predictive of the overall performance. The unchoked network is scale-free. There is no clustering except the initial stage.	[25]
Doing experiments on a testbed of 96 PCs	Random neighbor selection causes high operating cost	[26]

the last pieces problem; (2) the replacement of block level TFT with a bit level TFT solution does not necessarily yield better fairness; (3) the choke algorithm is fair and robust. Their recommendation is that the rarest first strategy already achieves excellent performance and there is no compelling need for more complex strategies. They do observe some scope for improvement in the choke algorithm. While this study sheds light on the important properties of the choke algorithm, however their measurement is from a single-peer view point which cannot evaluate the dynamics of BitTorrent. Also, the reasons behind the observed properties of the choke algorithm are unclear.

Bellissimo et al. [19] collected statistics from two trackers to conduct their measurement-based evaluation. The statistics included the user's random ID, how long the user had been connected to the torrent, and the number of bytes uploaded, downloaded, and remaining. Their results showed that (1) BitTorrent can serve large files effectively; (2) the incentive mechanisms are effective.

In [23], Skevik et al. focused on the effect of the firewalls and the time the peers have stayed in the system. They monitored the torrents of RedHat and Mandrake Linux for several months by executing a modified client and a crawler. They observed that peers behind a firewall have low bandwidth usage and they are more likely to leave the system early because of their poor download performance.

Karagiannis et al. [8] quantified the influence of BitTorrent on ISPs using two types of real measurement data, the BitTorrent tracker log for RedHat v9.0 and the payload packet traces collected using a monitor installed on the access link of the network of a residential university in USA with 20,000 users. While the paper considers the more general issues of peer-assisted content delivery and its impact on ISPs, we are specifically concentrating on their measurement based evaluation and observations about BitTorrent. They indicated that BitTorrent is locality-unaware which severely increases the ISPs' cost, typically resulting in the same content being downloaded into the same ISP multiple times from external peers. They also found that users requesting the same content have a 30%-70% overlap in their lifetimes, consequently helping each other in the download. An additional 20%-40% improvement in terms of cross-ISP downloaded content can be effected by giving incentives for users to stay around after they complete the download.

Andrade et al. [20] focused on measuring the cooperation in BitTorrent. In their measurement, they defined three metrics: (1) free-riding ratio, which is the percentage of free-rider peers, (2) seeding ratio, which is the percentage of seeds in a torrent, and (3) sharing ratio, which is the total uploaded data divided by the downloaded data. They evaluated the metrics by collecting logs in BitTorrent communities such as bt.etree.org, and piratebay.org. Then, they measured the downloading time for free-riders and non free-riders, and the amount of free-riding and low-sharing behavior in BitTorrent communities: etree and easytree. Their results demonstrate that BitTorrent's TFT protocol does discourage free-riding successfully. However, if there are a large number of seeds in the torrent, the TFT mechanism may not work effectively to prevent free-riding.

Legout et al. [24] conducted experiments on private torrents and collected data from peers in a controlled environment. They focused on the choking algorithm in BitTorrent to observe peers' individual behavior during the downloading process. Their experiments were performed on PlanetLab.

They collected logs of sent or received messages, and state changes using a modified version of BitTorrent. Their results reveal that the choking algorithm can cluster peers with similar bandwidth, reward contributing peers efficiently, and make peers have a high uploading utilization.

Dale et al. [25] looked at the topologies of different kinds of networks that can be clearly identified within a BitTorrent swarm. In particular they investigated the evolution of four kinds of networks that can be used to characterize a swarm: (1) Connection Network: the network of neighbors that each peer maintains. (2) Interest Network: the network defined by the interest that peers maintain in other peers. (3) Unchoked Network: the network that is formed by the peers and the neighbors that are unchoked by each peer. (4) Download Network: the network that is formed linking peers to other peers from which they are downloading. Among these networks, all networks are directed graphs except for the connection network which can be considered undirected. In their experiments, they used a modified BitTornado [27] client running on more than 400 nodes of PlanetLab that form the swarm. They found that:

- The initial stage is not predictive of the overall performance.
- The unchoked network is scale-free.
- There is no evidence of clustering in their experiment except in the initial stage.

Rasti et al. [21] pointed out that several of the measurement studies of BitTorrent were just based on observing the behavior of a few instrumented peers. This may not necessarily provide a representative view of the overall BitTorrent performance. Instead, a distribution of the performance indices among the peers provides a broader view of BitTorrent performance. Furthermore, the effect of different peer-level and group-level properties on the performance needs to be identified. To verify their theory, they analyzed a BitTorrent tracker log to estimate the distribution among peers in a torrent and identify the effect of the peer-level and group-level properties on the performance. They found: (1) the distribution of performance among peers is roughly uniform; (2) the TFT mechanism is the main factor that affects the performance. They could not observe any direct relationship between peer-level performance and main peer and group-level properties. Furthermore, the average upload rate of individual peers had the highest correlation with its observed performance [21].

Lei et al. [26] did an experiment to test the performance of BitTorrent's neighbor selection. Their experiment includes 96 PCs, each of the same configuration. They used BitComet as the BitTorrent client for their experiments. They observed that the random neighbor selection in BitTorrent does not take into account the communication cost, which will result in low transmission rate and high operating cost.

We summarize the various observations on BitTorrent performance obtained through measurements in Table II.

B. Simulation

Evaluating the impact of various factors on the overall performance is very difficult using measurement-based techniques. Simulation based studies on the other hand provide much greater flexibility to vary the various mechanisms used in BitTorrent and observe the impact through extensive experimentation. Compared with the measurement-based approach, the simulation-based approach has two advantages. One advantage is the greater flexibility in controlling the various configuration parameters of BitTorrent mechanisms. Another advantage is that it allows us to study the impact of variations in a particular mechanism while keeping the rest fixed, which is very difficult to achieve in measurement-based experiments. Simulation-based approaches are especially useful in topology related studies since a crawler does not yield information about a peer's neighbors in BitTorrent because of the lack of distributed mechanisms for peer discovery or lookup. Furthermore a peer's connectivity information is not shared with the tracker and hence tracker logs do not yield overlay information [31].

Many of the simulation based studies make simplifying assumptions for the sake of ease of simulation. Sometimes, these assumptions may not necessarily be realistic. Very often the underlying network topology is simplified and packet-level dynamics are not fully represented. Most of the simulators concentrate only on representing the behavior at the overlay network level. Thus the impact of the underlying physical network is not clearly reflected in the simulations. Also many simulations consider a smaller set of peers in a swarn than is normally encountered in practice. The results from simulation studies are summarized in Table III.

Felber et al. [28] presented a simulation based study of the effect of different peer and piece selection strategies on the performance of BitTorrent. In their peer selection strategy, they assumed that a peer has a global view of the state of all the other peers and thus selected the most suitable peer to exchange pieces with based on the peer selection strategy. They considered several peer selection strategies including random, and several variations based on the number of pieces possessed/missing in the other peers. Their results demonstrate that there is no clear winner among all the strategies in all scenarios. Similarly, they considered two different piece selection strategies: random and rarest. Among these the rarest strategy seems to deliver the most consistent performance across different peer selection strategies.

Bharambe et al. [29] evaluated the impact of BitTorrent's core mechanisms on overall system performance under a variety of flash-crowd workloads using a simulator, which can model not only peer's activity such as joins, departures, and block exchanges, but also BitTorrent's mechanisms like TFT and rarest first. To make the model simple, they did not model network propagation, the packet-level dynamics of TCP connections and the shared bottleneck links in the interior of the network. They focused on the link utilization, the fairness and the optimality to quantify the effectiveness of BitTorrent. They made their simulator named BTSim publicly available [34]. From the simulation results they found that:

- BitTorrent is robust and scalable, and ensures high uplink bandwidth utilization.
- The TFT policy fails to prevent the unfairness in BitTorrent.
- The rarest first policy is critical in ensuring that the newly joined peers have something to exchange with others.

TABLE III
SUMMARY OF SIMULATION-BASED STUDIES OF BITTORRENT PERFORMANCE

Simulation Features, Assumptions and Shortcomings	Results	References
Event-driven simulator with resolution of a millisecond. Peers have global view for peer selection	No appreciable impact of peer selection strategies. Rarest first piece selection strategy gives balanced performance	[28]
The event-driven simulator models both peer's activity and BT's mechanisms. Network propagation, the packet-level dynamics of TCP connection and the shared bottleneck links are not modeled	BT is robust and scalable at ensuring high uplink bandwidth utilization. The TFT policy fails to prevent unfairness in BitTorrent. The rarest first policy is critical to ensure the new joined peers have something to exchange with others.	[29]
Simple simulator contains one seed serving a file composed of n pieces. Network latency, connection costs or complex topologies not modeled.	By selectively uploading to its poorest neighbor, the peer ensures that the data remain in the network for the longest period, thus reducing its own burden.	[30]
The simulator is developed in MATLAB The authors simulated the tracker protocol in mainline client 4.0.2.	The overlay of BT is not a random graph. A large number of NATed peers decreases the robustness of the overlay to attacks significantly.	[31]
The simulator runs in rounds and each round lasts for 10 seconds A leecher sorts peers based on the data they offer The bottlenecks is downlinks/uplinks. The authors did not model any delay.	The peer set size (PS) and the percentage of outgoing connections (OC) have a significant impact on the BitTorrent's performance. Decreasing OC is more efficient than increasing PS. The choice of PS and OC can have an impact on the overlay structure.	[32]
The set of nodes involved in BT session is fixed. Endgame mode is not simulated. The neighbor set of each BT node is static. Delays are not simulated except network packet transmission delay.	The current BT is far from the optimal solution that minimize the end-to-end delay of distributing a file from a source to multiple receivers. The peer selection strategy may lead to mismatched peering and make BT deviate more from optimum.	[33]

The simulations were restricted to a peer set no larger than 15 peers, while the real implementations can have up to 80. This might influence some of the results since the accuracy of the piece selection strategy is affected by the peer set size.

Adar [30] demonstrated through simple simulations that a peer possessing most of the file pieces can significantly reduce its effort in servicing other peers while at the same time improving network performance. They simulated the effect of bit welfare. There is only one seed serving a file composed of n pieces in the simulator and three seed strategies are considered: Random, in which the seed chooses the peer to give piece randomly, Preferred, in which a peer is chosen to receive pieces from a preferred subset, Poorest-first, in which the peer who has the fewest pieces is chosen. Based on their experiments, they found that the Poorest-first runs better than other algorithms. The experimental results showed that by selectively uploading to its poorest neighbor, the peer ensures that the data remain in the network for the longest period, thus reducing its own burden.

Urvoy-Keller et al. [32] adopted a simulation approach to evaluate the impact of the overlay topology parameters on the BitTorrent performance. Their results show that the two parameters, the peer set size (PS) and the percentage of outgoing connections (OC), have a significant impact on the BitTorrent's performance. In addition, decreasing OC is more efficient than increasing PS. Furthermore, the overlay structure can be affected by the choice of PS and OC.

Al Hamra et al. [31] followed up on the work by Urvoy-Keller et al. [32]. They specifically concentrated on evaluating the properties of the distribution overlay of BitTorrent. They presented an in-depth study of the overlay topologies in BitTorrent by conducting simulation experiments using a simulator developed in MATLAB. They analyzed the relation between the overlay properties and the BitTorrent performance considering the following four parameters: average peer set size, the time for a peer to reach its maximum peer set size, the diameter of the overlay, the robustness of the overlay to attacks and high churn rate. They pointed out that a large peer set size can make BitTorrent more efficient. More importantly, they showed for the first time that the overlay in BitTorrent is not a random graph, and connectivity of a peer to its neighbors depends on its arriving order. In addition, they showed that a large number of NATed peers (peers behind a NAT or a firewall) significantly decreases the robustness of the overlay to attacks.

Wu et al. [33] were specifically interested in answering the question of how close BitTorrent is to the optimal solution. They proposed a centrally scheduled file distribution (CSFD) protocol that can minimize the total elapsed time. In addition, they compared the performance of BitTorrent and CSFD. They found that the current BitTorrent is far from the optimal solution that minimizes the end-to-end delay of distributing a file from a source to multiple receivers. Moreover, the peer selection strategy of BitTorrent may lead to mismatched peering, and make BitTorrent deviate from the optimum.

C. Analytical Modeling

Several analytical models of BitTorrent-like file sharing system have been proposed in the literature (See Table IV). Existing analytical studies can be classified into two categories: homogenous, where all peers have the same upload and downloading rate; and heterogeneous, where peers have different uploading and downloading rates. Analytical models are clearly able to reflect the effects of different parameters on BitTorrent performance. They permit efficient and detailed exploration of the parameter space to evaluate the effect of variations of not just a single parameter, but also the combined effect of variations of several parameters. However, many of these models are based on sometimes unrealistic assumptions like peers having global information about the state of all peers, simplifying assumptions on the underlying network topology, and on the arrivals and departures of peers. Most of

TABLE IV
SUMMARY OF ANALYSIS-BASED STUDIES OF BITTORRENT PERFORMANCE

Focus of Analysis	**Modeling Approach**	**Results**	**References**
Service capacity and fairness	Homogenous Branching process, Markovian	P2P systems achieve favorable scaling in terms of average download delay with increasing load. ε-peerwise fairness is a practical approach, for example as used in BitTorrent.	[35]
Scalability and incentive mechanisms	Homogenous Fluid-Flow	The system scales well with the number of peers. The incentive mechanisms are efficient. Optimistic unchoking may encourage free-riders	[3]
Single-Torrent system, file availability Multi-torrent system performance	Homogenous Fluid-flow	BitTorrent is good at dealing with "flash crowds." Seed departures and decreasing peer arrivals causes file to become unavailable soon. Peers with higher downloading capacity do not upload as much as slower peers. Inter-torrent collaboration is effective to improve performance	[36]
The distribution of the individual chunk	Homogenous	The modified routing policies perform better	[37]
Evaluation of file distribution time Application-level differentiated service model	Heterogeneous Fluid-Flow	Obtains an explicit expression for the minimum achievable time to distribute a file.	[38]
Influences of free-riding	Heterogenous Fluid-flow	BitTorrent mechanism can prevent free-riding successfully	[39]
The distribution of the peers' download state, file availability and system death	Homogenous Fluid-flow, Markovian	Peer distribution in terms of their current download state follows a U-shaped curve, with most peers in either just starting the download or possessing the entire file. The seeds' departure rate and leechers' abort rate can influence the peer distribution. The TFT strategy cannot improve the file availability or prevent system death.	[40]
The download behavior and incentive mechanism	Homogenous Queueing Model	First attempt at a queueing model for BitTorrent protocol	[41]
Altruism and Incentive Mechanisms. Focus on reciprocation and upload bandwidth sharing	Homogenous Probabilistic model	Altruistic contribution by high-capacity peers seem to cause the most benefit in download performance. TFT policy is not the major cause for good performance	[42]
Effect of Bandwidth Heterogeneity on download performance	Heterogeneous Fluid-flow	Heterogeneous bandwidth can have a positive effect on content propagation	[43]
Two-class problem. Focus on Service differentiation among the classes. Bandwidth diversity problem	Heterogeneous Fluid-flow	System of differential equations admits a unique stable equilibrium point. Whether the system of differential equations has a stable state or not depends on the initial conditions.	[44]
File downloading time File availability Both Single class and Multiclass	Heterogeneous Fluid-flow	Stochastic differential equation approach, yields closed form solutions for several performance measures including average download time, system throughput, number of peers and seeds. Sensitivity analysis of performance measures.	[45]
File dissemination in P2P networks, Minimum file distribution time	Heterogeneous deterministic	Distribution time of any natural random strategy as adopted by BitTorrent is proportional to the optimal time	[46]
Heterogeneous peers and performance using simple models using balance equations.	Heterogeneous deterministic flow balance equations	Simple model yields accurate results, verified with simulation. Token based scheme, which can yield better performance for low-bandwidth peers by giving higher weightage to their upload contribution.	[47]

the analytical models rely on a fluid-flow approach to characterize the system behavior. In particular, the models represent the evolution of the system by tracking the number of peers, $X(t)$, and the number of seeds, $Y(t)$, either as a function of time, or in steady-state. The evolution is represented using (stochastic) differential equations involving various system parameters. Typical parameters that characterize the system evolution include:

- λ, the arrival rate of new requests. The interarrival time is usually assumed to be exponentially distributed.
- μ, the upload bandwidth of a peer.
- c, the downloading bandwidth of a peer.
- θ, the rate at which peers abort their download
- γ, the rate at which seeds leave the system, and
- η, the effectiveness of file sharing.

Some models keep track of the different stages of downloading that a peer may exist at any point of time. This typically involves dividing the peers, $X(t)$, into different subsets, $X_1(t), X_2(t)$, etc. each representing the different stages of download. However, this increases the number of simultaneous equations that need to be solved and hence the complexity of the model.

Yang et al. [35] modeled and analyzed P2P systems in terms of their "service capacity" for both transient and steady-state regimes. They considered an abstract model of P2P file sharing system, and focused on peers that are sharing a single file. They concentrated on the file download process and assumed that each peer knows the address and file availability at other peers. Furthermore, they assumed that no peer leaves the system before finishing their downloading. They first presented transient analysis of the P2P system using both deterministic and branching process approaches to determine the average download delay. They demonstrated that P2P systems achieve favorable scaling in terms of the average download delay with increasing load. They also show that parallel uploads from a peer, where the upload bandwidth is used by a peer to simultaneously upload to multiple peers, is effective even when peers are uncooperative. Then they modeled the stationary regime using a Markovian approach. Furthermore, they studied the fairness issue in P2P systems by considering different notions of fairness for service allocation in the stationary regime. Achieving global proportional fairness and pairwise proportional fairness is not easy for robust implementation. They then proposed an ε-peerwise fairness which is better suited for a dynamic environment with peers joining and leaving the system. This is the strategy used in BitTorrent with its choking algorithms.

Qiu et al. [3] presented an analytical model of BitTorrent based on a fluid flow model. Their work was influenced in part by the earlier work of Yang et al. [35]. In developing their model, they made several assumptions: downloaders become seeds with a probability that is determined by parameters like the number of peers; downloaders can abort their downloading after a certain time that is exponentially distributed, and each seed stays in the system for a random time which is also exponentially distributed. They developed a simple deterministic fluid-flow model to describe the evolution of the number of leechers and the number of seeds, in terms of the arrival rates, the upload and download bandwidths, the abort rate of downloaders, the rate at which seeds leave the system and the effectiveness of file sharing. They used this model and studied the steady-state performance, with special attention to the protocol's scalability and efficiency. They found that the average downloading time is not related to the peer arrival rate, and hence the system scales very well with the number of peers. Their model clearly demonstrated that the average downloading time decreases with efficient file sharing and increases with increasing seed departure rate, as expected. Also changes in the uploading and downloading bandwidth impact the downloading delay to some extent. They then presented a game-theoretic analysis of both the choke and rarest first algorithms. They show that the incentive mechanisms in BitTorrent are efficient. They also show clearly that the optimistic unchoking might lead a free-rider to get upto 20% (In general, $1/(n_u + 1)\%$, where n_u is the number of regular unchokes) of the maximum downloading rate. They also presented experimental results from simulations to validate the results from their models. The assumption about global knowledge of all peers for peer selection is a major limitation of this approach.

Guo et al. [36] followed up the idea of [3]. First they analyzed the file downloading trace files obtained from two dedicated tracker sites that host multiple torrents. They also analyzed BitTorrent metafile downloading traces that were collected from a large commercial server farm hosted by a major ISP and a large group of users connected to the Internet. From the trace analysis, they concluded that the peer arrival rate to a torrent decreases exponentially. They studied the torrent lifespan, the duration from birth until the file becomes unavailable. They derived the expressions for the torrent lifespan and the downloading failure ratio, the ratio of peers that do not complete the download to the total peer population for a torrent. They extended the fluid-flow model in [3] using the model for the exponentially decreasing peer arrival rate to construct the torrent evolution model. They found that both peers and seeds increase exponentially initially, but then decrease exponentially at a slow rate. The seed departures result in the torrent being unable to keep up with the peer service demand, causing the torrent to eventually die. The random arrival and departures of peers and seeds causes the performance to fluctuate significantly over the torrent lifetime, especially for small torrents. Furthermore, they demonstrated that the biased service policy of seeds towards peers with higher download bandwidth might result in these peers not contributing their fair share of upload capacity to the torrent. They also proposed a graph based multi-torrent for analyzing inter-torrent collaborations. Their analysis demonstrated the feasibility of using multi-torrent collaboration effectively to improve performance.

In [37], Arthur et al. proposed a BitTorrent-like model using which they analyzed the data disseminating protocols of BitTorrent and related P2P networks. In this model, they ignored the TFT strategy and assumed that all nodes have the same bandwidth. In particular, they modeled the routing of data blocks on a directed graph over discrete time steps. They also analyzed some of the multiple network topologies and routing algorithms.

Kumar et al. [38] presented an analytical model of peer-assisted file distribution time in P2P networks using a fluid-flow based model. Their aim is to study the advantages of a peer-assisted file distribution system vis-à-vis a traditional client-server distribution system, specifically the scalability and efficiency of the new approach. In particular, they were interested in finding the minimum distribution time for the file to be distributed from the seeds to all the leechers in a general heterogeneous peer-assisted file distribution system. In their model, they made two assumptions: The bandwidth bottlenecks are in the uploading and downloading link rates instead of in the Internet core, and every node participates in the file sharing until it completes the file download. Using this model they obtained an explicit expression for the minimum distribution time. This expression explicitly shows the relationship of the distribution time to the file size, the seeds' uploading rate, and the leechers' uploading and downloading rate. They also point out that although their approach makes an unrealistic assumption about the flow of bits through a node compared to the chunk-based models that were proposed earlier, the resulting expressions provide a very good (lower-bound) approximation for the file distribution time. However, it is important to note that peer heterogeneity is better handled by their approach.

In [39], Yu et al. focused on analyzing the influence of free-riding in BitTorrent System. They extended the model in [3] with two classes of peers: nonfree-riders and free-riders. Their results show that BitTorrent's mechanism can guard against free-riding successfully. In addition, they discussed the system death induced by free-riding and pointed out that retaining a copy of each piece in the system can prevent system death.

Tian et al. [40] extended the fluid-flow model in [3] to study the distribution of the peers in different states of the downloading completion. Towards this end, they explicitly represent the peers as belonging to one of several states, S_0 through S_N, where $1/N$ is the level of granularity at which they view the current download progress. The peer state behavior is now viewed as a continuous time Markov chain. From this they derived the distribution of the peers in each state of the download. The distribution is observed to be a U-shaped curve. From the model, they found that seeds' departure rate and leechers' abort rate can influence the peer distribution. Furthermore, the TFT strategy cannot improve the file availability or prevent the system death caused by a sudden departure of a finished peer.

Saraswat et al. [41] presented a queuing model for BitTorrent using the Java Modeling Tools [48]. In their model, each BitTorrent client is modeled as a load dependent station. The stations' service request time corresponds to the time taken to download a piece, the service time correspond to the inverse of the downloading bandwidth, and the number of jobs in the queue represents the total number of pieces downloaded. They set the service time dependent upon the number of jobs in the station to ensure that the incentive mechanisms work well. They focused on the download behavior and incentive mechanisms.

Piatek et al. [42] presented a model for the altruism in BitTorrent. They made the following assumptions: (1) The distribution of a typical swarm may not be identical; (2) The active set sizing is uniform; (3) There is no steady state; (4) Swarm performance is limited by upload bandwidth. The authors built the model based on the expressions for the probability of TFT reciprocation, expected downloading rate, and expected upload rate. In the model, they considered two definitions of altruism. It can be defined as the difference between expected upload rate and the download rate. Another definition is any upload contribution that can be withdrawn without loss in download performance. Their result shows that the TFT policy is not robust, and the altruistic contribution plays an important role in BitTorrent's performance.

The first heterogeneous fluid model of BitTorrent-like file sharing system was proposed by Lo Piccolo et al. in [43]. To assess the effect of different access link capacities on BitTorrent-like file sharing system, they developed a fluid model with access links of two different capacity classes by extending the fluid-flow model presented in [3]. To keep the heterogeneity simple, they only considered links of two different link rates, viz. a high speed downloader and a low speed downloader. Furthermore, they assumed that the links are symmetric, i.e., the upstream rate is the same as the downstream rate. They analyzed the effects of bandwidth heterogeneity on file transfer dynamics and content diffusion process in detail. In particular, they compared the performance of heterogeneous networks and the equivalent homogeneous networks under different conditions of equivalence. Their results show that heterogeneous bandwidth can have a positive effect on content propagation among peers if appropriate criteria were chosen.

Clévenot-Perronnin et al. [44] argued that the model in [3] considers only homogeneous peers, but in practice, peers have diverse bandwidth characteristics, such as Ethernet access, dial-up modem access and broadband access. This limits the model's applicability. Therefore, they presented a general multiclass model for heterogeneous peers with different access bandwidth and multiple differentiated service classes. In particular they considered in detail a system with two classes of peers, distinguished by their different upload and download speeds. In their model, they allow an uploader (either a seed or peer) to statically allocate its upload bandwidth to the two classes of peers. They used this model to analyze two problems: service differentiation and bandwidth diversity. For the service differentiation problem, they described that the system of differential equations admits a unique stable equilibrium. For the bandwidth diversity problem, they indicated that whether the system of differential equations has a stable state or not depend on the initial conditions.

Fan et al. [45] extended the model in [3] to obtain a fluid-flow model based on the stochastic differential equation approach. In this model, they divided peers into three types: leechers that have a few chunks, leechers that have most chunks, and seeders. They define the connection probability $\rho(\leq 1)$ as the probability that a peer maintains connectivity with another peer in the torrent. They analyzed the file downloading time and file availability in BT-like systems with this model. They obtained closed-form solutions for the average number of seeds and leechers, the average downloading time and the steady state throughput of the system. The closed-form solutions show the effect of various system parameters including the peers' arrival rate, seeds' departure rate, connection probability and transmission bandwidth on the performance measures explicitly. They also carried out sensitivity analysis of the performance metrics with respect to the different parameters. They used a discrete-event simulator to validate the results obtained from analysis. They then extended the model to account for peers situated behind firewalls and discussed the impact on the system performance. Their results clearly show that those peers not behind firewalls play an important role in determining the overall system performance. They also discussed the effect of different chunk selection algorithms at the peers on the file availability within the system. Through mathematical analysis they showed that the rarest first policy indeed is the best to increase file availability. In practice this is true for low connection probability among peers. However when ρ is large, this may cause a reduction in file availability. To improve file availability, they proposed a file availability enhancement algorithm which randomizes the selection of the chunk to download, but still giving preference to the rarest chunk with higher probability. They showed that this enhancement improves the file availability in all circumstances.

Mundinger et al. [46] studied the file dissemination in P2P networks using the 'uplink sharing' approach, which is a variation of the classic broadcasting problem. In their model, nodes have different upload capacities. They assumed that the available upload capacity is shared equally amongst the connections, and uploads are not interrupted until complete, which means the rate is always positive. They showed that a simple and natural random strategy adopted by BitTorrent results in a distribution time which is proportional to the optimal time achievable.

Liao et al. [47] considered heterogeneity in a BitTorrent system by distinguishing users into two classes, viz. high bandwidth users who have high upload-link capacity, and low bandwidth users with low upload-link capacity. They proposed a mathematical model to predict the average file download delay for the two classes of clients. This model represents the system behavior in steady-state by matching the amount of data uploaded and downloaded by the peers. In addition, they designed a token-based TFT scheme. In this scheme tokens are used as a means for deciding how a peer uploads data to its neighbor. Tokens are earned by a node by uploading to its neighbor. A node keeps track of the number of tokens for its neighbors and makes its upload decisions based on the tokens. By using a biased means for a low-bandwidth peer to earn more for uploading to a high-bandwidth peer, the authors design a system which ensures better download performance for low-bandwidth peers, albeit with a higher upload contribution by the high-bandwidth peers. They validate their model by comparing the results with simulation results for several experiments conducted using the BTSim simulator provided by [34] for different scenarios. The results show that the proposed mathematical model is quite accurate in predicting the file download performance.

D. Summary and Comparison of Performance Studies

In this section, we examine the performance studies from the perspective of the different aspects of the BitTorrent protocol that were studied. We summarize our findings in Table V, in particular highlighting how the different aspects were studied using the different performance evaluation methods, with more or less the same conclusions.

All the three performance evaluation methods have focused on evaluating the piece exchange mechanisms in BitTorrent. The general findings from the different approaches are in agreement with each other. In particular, it is found that the TFT mechanism is very useful in ensuring co-operation among the peers. Furthermore, the general observation is that the presence of altruistic peers encourages free-riding. Similarly, the rarest first policy has been found by all methods to be very effective in increasing download performance and decreasing the file download time.

Topology related studies are feasible only through measurement and simulation, since these methods explicitly reflect the effect of the topology on performance. In particular, the topology's effect is direct in measurement. In simulation studies, the topology is usually abstracted at the overlay topology level. Analytical modeling does not explicitly track the topology, but its influence may be reflected through system parameters like the connection probability ρ in [45]. References [25] and [24] studied the BitTorrent topology using measurement-based studied, while [31] and [32] studied the same topic by simulation. They observed the overlay structure and evaluated the parameters which can impact the overlay topology. In addition, BitTorrent's neighbor selection and the impact of BitTorrent on ISP traffic were also measured by [8] and [26]. They show that the random neighbor selection and BitTorrent's locality-unawareness have a significant impact on the overall performance.

IV. Improvements to BitTorrent Mechanisms

The phenomenal success of BitTorrent has in its wake attracted the attention of several researchers who examined its performance in detail as highlighted in the previous section. Several researchers subsequently suggested improvements to existing BitTorrent mechanisms, or suggested new mechanisms to replace existing BitTorrent mechanisms with the aim of further improving its performance. In general, the suggested improvements can be viewed as belonging to two categories: (a) mechanisms that suggest improvements to BitTorrent's overlay topology formation and maintenance, and (b) modifications to the piece exchange mechanisms in BitTorrent. In this section, we present an overview of the research literature suggesting improvements to BitTorrent. We then categorize these mechanisms as stated above.

A. Improvements to Overlay Topology

First, we review several proposals that aim to improve the performance of BitTorrent by changing the way peers establish their connections to other peers while joining, and thereafter while maintaining the connections. These techniques are aimed at addressing the topology mismatch problem between the overlay and the underlying physical network. The techniques are based on using the location of the peers to optimize the overlay topology. We classify the techniques into two categories: (1) proximity aware techniques, and (2) ISP-friendly techniques.

1) Proximity Awareness: BitTorrent builds its overlay network by selecting neighbors randomly. The randomized selection results in a significant divergence between the logical overlay network and the underlying physical network infrastructure [49]. This causes wastage of network resources and potentially results in suboptimal downloading time. For instance, two neighboring peers in the overlay may be in different countries far away from each other. In addition, selecting neighbors randomly also increases the probability of having capacity-limited peers connected to high-capacity peers. To address these problems, some researchers have explored the method of using proximity in the topology construction and maintenance in BitTorrent. The major aims of exploiting proximity are achieving an efficient usage of network resources, and reducing the individual downloading time for the peers.

Qureshi [49] was among the first to suggest the use of proximity in the BitTorrent overlay network. He pointed out that the present overlay construction algorithms ignore the problem of matching the overlay network and the underlying

TABLE V
SUMMARY OF METHODS USED IN STUDYING BITTORRENT PERFORMANCE

	Issues	Measurement	Simulation	Analysis
Piece Exchange	TFT(Incentive mechanism)	The TFT policy works well [18]. The replacement with a bit level TFT solution is not appropriate [22]. The TFT mechanism increases the cooperative behavior and discourages free-riding, but if there are a large number of seeds, it may not work effectively to prevent free-riding [20].	The TFT policy fails to prevent unfairness [29].	The incentive mechanism are efficient [3]. Optimistic unchoking may encourage free-riders [3]. BT's mechanisms can guard against free-riding successfully [39]. TFT policy is not the major cause for good performance; it is the altruistic contribution by high-capacity peers [42]
	Piece Selection	The rarest first policy works well [18]. The rarest first policy prevents the reappearance of rare pieces and avoids the last pieces problem [22].	Rarest first piece selection strategy gives balanced performance [28]. The rarest first policy is critical to ensure the new joined peers have something to exchange with others [29].	Rarest first strategy is still the best way to increase file availability [45].
Overlay Topology	Topology	Choking algorithm can cluster peers with similar bandwidth [24]. There is no evidence of clustering except the initial stage [25].	The overlay is not a random graph [31] The overlay topology can be affected by the choice of peer set size and the percentage of outgoing connections [32].	
	Neighbor Selection	The random neighbor selection in BT causes low transmission rate and high operating cost [26]. BT is locality-unaware, which increases the ISPs' cost [8].	The peer selection strategy may lead to the mismatched peering and make BT deviate more from the optimum [33].	
	Impact of Firewall/NAT	Peers behind a firewall have low bandwidth usage [23]. Peers behind a firewall often disconnect early [23].	A large number of NATed peers decrease the robustness of the overlay to attacks significantly [31]	Presence of non-firewalled peers essential to good system performance [45].

network. He suggested that peers that are close by in the real world should be close by in the overlay network. Formally, for any two nodes A and B, he specified their round-trip communication latency as $rtt_{[A,B]}$. Given three nodes A, B and C, if $rtt_{[A,B]} < rtt_{[A,C]}$, then $P(A,B) > P(A,C)$, where $P(A,B)$ is the probability that A and B are peers in the overlay network. Several techniques including using network coordinates coupled with a probabilistic flooding algorithm were exploited in the topology construction and maintenance. This was done in two phases: (1) estimating the node's network coordinates; (2) using a gossip-like protocol for near neighbor discovery. The expected result was that an overlay network that closely matches the underlying network topology in terms of node proximity should yield shorter downloading time and more efficient usage of network resources.

Zhang et al. [50] analyzed location proximity in BitTorrent system not only based on topology proximity, but also for cooperative proximity. They indicated that both neighbor selection and piece selection phases are proximity-unaware. This can cause problems such as topology mismatch, and long average downloading time. They considered a proximity aware topology construction algorithm. Their approach consists of two steps. In the first step, they give each node a synthetic network coordinate by using the algorithms in [51]. In the second step, each node selects its neighbors by their Euclidean distance. Their simulations are conducted using a modified version of BTSim [34] extended to exploit proximity in overlay topology construction. They built a 3-dimensional space representation of the network, where each node's location was specified as a 3-tuple representing the three coordinates, which could be x-coordinate, y-coordinate and z-coordinate. When a node joins into the network, the coordinates will be assigned to it by the tracker. Then, the node will measure the Euclidean distance between other nodes and itself, and proactively select nodes with the shortest Euclidean distance as its neighbors. Their results show that using location proximity in BitTorrent can decrease the file downloading time about 11.3% for 1000 nodes torrent, and improve the network utilization, especially for moderate size torrent.

Koo et al. [52] proposed a neighbor selection strategy for hybrid P2P network like BitTorrent that favors picking neighbors with the most mutually disjoint content. In [52], they modeled peers as an undirected graph. In addition, they presented a genetic-algorithm-based method for neighbor selection. They use ns-2 [53] package to simulate the system. In their simulation model, there are three classes of peers with different bandwidths. They compared the proposed neighbor selection strategy with the existing strategy using two metrics: system throughput and average downloading time. Their results show that the new neighbor selection algorithm can improve system performance by increasing the content availability for peers from their neighbors.

The proposed methods of neighbor selection by proximity [49], [50] are still oblivious to ISP boundaries. Peers selected with proximity in mind may belong to different ISPs, thus still not addressing the cross-ISP traffic problem.

2) ISP-Friendliness: The wide use of BitTorrent has put ISPs in a dilemma [54]! On the one hand, BitTorrent, with a large number of users, constitutes an important source of revenue for ISPs. On the other hand, BitTorrent generates a large amount of cross-ISP traffic and makes the ISPs' cost increase significantly since BitTorrent's implementations ignore the underlying Internet topology and ISP costs. Lei et al. [26] mentioned that BitTorrent's fast downloading experience comes at a large cost to the network in terms of resources. This may consequently affect other Internet services supported by the ISPs.

To address the cross-ISP problem, Bindal et al. [9] designed a new algorithm based on biased neighbor selection, in which a peer chooses its neighbors mostly from peers within the same ISP, and only selects a few from the outside its ISP domain. This is because peers in the same ISP are highly connected. However, some connectivity to peers outside the ISP boundaries needs to be maintained in order to obtain new blocks. They indicated that the biased neighbor selection can be implemented either by changing the tracker and client or by employing P2P traffic shaping devices. When there is a request from a peer for neighbor list, the tracker will choose k external peers and $35-k$ internal peers to feed back to the requesting peer. If there are less than $35-k$ internal peers, the tracker will ask the requesting peer to contact it again after certain duration. The tracker can know the peers' ISP location either by using the AS mapping or IP address to identify the ISP, or add a new header which contains the locality tag to HTTP proxy. Biased neighbor selection can also be implemented by P2P traffic shaping devices, which are situated alongside the edge routers of the ISPs and use deep packet inspection to identify P2P traffic. To understand the process of neighbor selection, the authors designed a discrete-event simulator and evaluated it over a network configuration with 14 ISPs. They considered the flash crowd phase. The main metrics evaluated include the download time, and ISP traffic redundancy which is the average number of times the blocks of the downloading file travels into the ISP until all peers inside the ISP finish their download. The simulation result shows that biased neighbor selection can reduce the cross-ISP traffic and maintain the nearly optimal performance of BitTorrent.

Yamazaki et al. [55] proposed a series of strategies to reduce ISP costs called Cost-Aware BitTorrent (CAT). In CAT, the peers first acquire ISP cost information. Thereafter, path costs between any two peers can be computed using the ISP costs. The tracker selects peers in the list of peers it returns based on the shortest cost to the requesting peer. The inter-peer cost is used by a peer in the selection of candidate peers from which to download and upload pieces. The strategy is to minimize the cost incurred in the transactions, consequently minimizing the ISP costs. Through simulations for different scenarios they show that CAT can reduce the ISP cost and improve the performance.

B. Improvements to Piece Exchange Mechanism

As already mentioned in Sec. II, the tit-for-tat (TFT) mechanism ensures that peers contribute as much as they download. Similarly, optimistic unchoking (OU) enables new peers to get started, and enables peers to discover better matched peers for cooperation. In this section, we review several suggestions by authors to improve the piece exchange mechanisms in BitTorrent to further improve download performance, punish free-riders and address unfairness. The first set of mechanisms aims to promote collaboration among peers using the underutilized download bandwidth of peers to help others. The second set of mechanisms address the issue of free-riding through several approach to identify and punish free-riders. The third set of mechanisms address the issue of unfairness in BitTorrent by suggesting improvements to the unchoke mechanisms.

1) Collaboration among Peers: BitTorrent's piece exchange mechanism is designed to enforce fairness among peers through the TFT mechanism. Garbacki et al. [56] pointed out three limitations of TFT: (1) the newcomers are bootstrapped at the bandwidth cost of the existing peers; (2) no incentives for seeding; (3) peers with asymmetric Internet connections cannot fully use their downloading links since they are forced to download at the speed of their uploading link.

Garbacki et al. [57] proposed a protocol named 2Fast which extended the bartering model of BitTorrent. In 2Fast, a peer with idle bandwidth can join a download in progress as a helper to assist a peer (hereafter named the *collector*) and fetch missing fragments of the file being downloaded and thereafter upload them to the collector. A collector recruits peers that are willing to act as its helpers with the promise that the bandwidth contributed by the helper will be returned in the future. The authors used social incentives to enforce the delivery on the promise to return bandwidth in the 2Fast system. In [58], the same authors exploited social phenomena like friendships to enable collaboration. Here a collector finds helpers just like people collaborate with friends in communities. Through rigorous analysis, the authors show that adding a new helper will not help if the collector's downloading bandwidth is already saturated. They computed the minimum number of helpers that fills the collector's bandwidth. Finally, they conducted experiments to compare the BitTorrent protocol enhanced with the 2Fast protocol against the standard BitTorrent protocols using several metrics such as the downloading speedup, which is the ratio between the downloading time of a peer acting on its own and the downloading time of a collector supported by its helpers. The results show that the use of 2Fast protocol improves the download speed by up to a factor of 3.5 in comparison to the standard BitTorrent without helpers. Fig. 3 depicts the collaboration between a collector and its helpers in the 2Fast protocol.

Subsequently, Garbacki et al. [56] extended the 2Fast protocol and proposed a novel mechanism in which incentives are built around bandwidth rather than content. Bandwidth is unrelated to the interests of peers, so it is more suitable to be a trading unit. In the bandwidth-exchange incentive mechanism, helpers (peers) can use their idle bandwidth to help others with their download, regardless of the content. The new protocol is named amortized tit-for-tat (ATFT). In this protocol peers choose the bandwidth borrowers with the highest chance of returning the borrowed bandwidth. ATFT employs the exploration and selection operations to select peers with the highest contributions as the borrowers. Exploration extends the borrowers set by opening opportunities for bandwidth exchange, and selection reduces the borrowers set

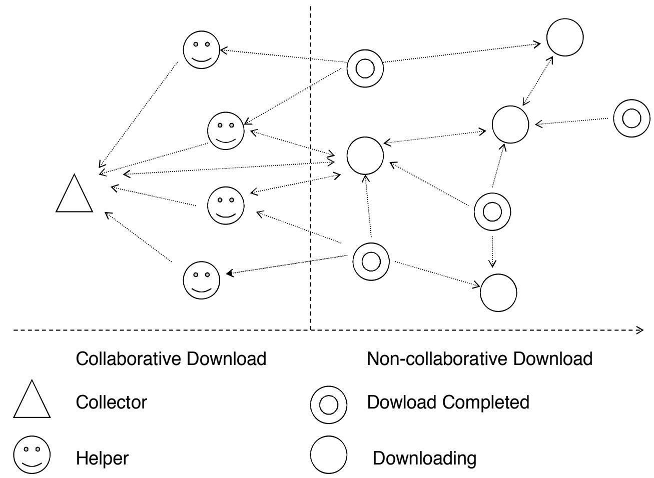

Fig. 3. Collector and helpers in 2Fast system

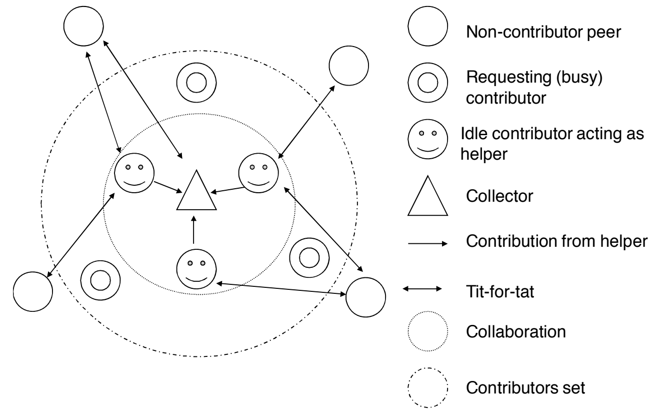

Fig. 4. The Collector and helpers in ATFT protocol

by removing the least promising peers. Fig. 4 illustrates the ATFT protocol from the view of the collector. The authors pointed out four benefits of ATFT. Firstly, newcomers pay the cost of bootstrapping by themselves. Secondly, ATFT provides an effective incentive for content seeding. Thirdly, peers can fill their downlinks completely with the helpers' bandwidth. Finally, it can be used to provide a certain level of anonymity. They conducted several simulation experiments using ATFT. In their simulations, a peer maintains a single value for keeping track of the bandwidth obtained from contributors to account for bandwidth contributions. The results show that ATFT improves the average download bandwidth of a peer by a factor of 2 to 6.

The above protocols do not explicitly specify how many pieces the helpers should download to make the system work well. On one hand, if helpers download too many pieces, it would incur excessive transmission cost; on the other hand, if the helper downloads too few pieces, it may not be very useful. Wang et al. [59] showed that helpers can work as effectively as seeders if they download only a tiny fraction of the file. Helpers in [59] are resource-rich nodes with spare uploading bandwidth that can be used to increase the total system uploading bandwidth and hence easing the bottleneck. They proposed an efficient strategy for utilizing the spare uploading capacity, where a helper joins a swarm just like a regular peer, and only downloads a small number, k pieces, of the file. Once a helper downloads k pieces, it will stop downloading, and at this time becomes a *microseed*. The microseed contacts the tracker and gets a list of peers who need help. The microseed implements choking and unchoking algorithms. Once the microseed knows that a neighbor has already downloaded all the pieces that it possesses, it will disconnect from the neighbor. Helpers download only a small number of pieces because they are not interested in the file. Likewise, downloading a large part of the file is a waste of helpers' bandwidth. The authors used a fluid model based on the model presented in [3] to analyze the system performance. They made several assumptions: the peer arrival process is Poisson; peers are homogeneous with the same uploading and downloading bandwidth; peers download at the same rate when they are downloading. They also implemented a simulator for the original BitTorrent and BitTorrent with helpers, using which they compared the average downloading time. They presented several conclusions: (1) helpers are almost as effective as seeds though they only download a tiny fraction of the file; (2) helpers that download too many pieces will hurt the system performance; (3) average downloading time of a BitTorrent system with helpers is much shorter than that of the BitTorrent system without helpers; (4) it is more effective for BitTorrent to have a lot of helpers than to have a few extra seeds.

2) Addressing Free-Riding: A free-rider is a node that downloads pieces from other peers but does not upload any pieces to others. Conventional P2P systems lack incentive mechanisms and are vulnerable to free-riding. Incentive mechanisms encourage cooperation among peers by rewarding contributors and punishing free-riders. Although BitTorrent has incentive mechanisms such as TFT, many researchers pointed out that they are not effective in preventing free-riding.

Jun et al. [60] evaluated the original incentive mechanism of BitTorrent using measurements on a P2P network set up on PlanetLab [61]. They view the peer behavior in BitTorrent as akin to the iterated prisoner's dilemma. They found that free-riders are not effectively punished, and peers that contribute to others are not rewarded appropriately. To address this problem, they proposed a new mechanism which is more robust against free-riders. In their new mechanism, they defined peers' upload amount u and download amount d for each link. Then, they define a concept named deficit, which is $u-d$. A peer ensures that the deficit is always upper-bounded by the formula: $u - d \leq f.c$, where the constant c is the size of a fragment, and $f(\geq 1)$ is called a nice factor. The factor determines the amount that a peer is willing to risk for a chance to establish cooperation. Peers upload evenly to all links as much as they can under this condition. The authors evaluated the new mechanism experimentally through measurements using PlanetLab. The experiments consisted of one tracker, one seed and 170 leechers running BitTorrent 4.0.0. All of the leechers begin to download at the same time. During the experiment, the authors evaluated two metrics: the downloading time and the uploading amount. The results show that the original incentive mechanism in BitTorrent is suboptimal and the proposed mechanism is more robust at preventing free-riding.

Sirivianos et al. [62] presented a new free-riding technique named the large view exploit. In this approach they modified the BitTorrent client to: (a) not upload content to others, (b) periodically contact the tracker behaving as a new peer and obtain a new list of peers, and (c) establish connections with peers in the new list, thus exploiting their optimistic unchoke contributions. The authors performed several experiments both with PlanetLab resident torrents and public torrents to study the effectiveness of the large view exploit. Their results show that the modified peer can easily achieve better performance than the traditional peers. The authors then suggest a modification to the BitTorrent tracker and clients to address this problem. The aim is to give the clients and the tracker a consistent view of the whole swarm. The approach is to use a peer's IP address together with a pseudo-random number sequence to enable other peers to attempt to identify free-riders. The modification requires the tracker to perform a larger amount of housekeeping, and may impact the scalability of the approach. The authors are continuing to investigate this approach further to improve scalability.

Chow et al. [63] presented a novel approach from the perspective of using the seed capacity appropriately with the goal of reducing the free-riders. Several of the measurements and simulation studies of BitTorrent reveal that the leechers' downloading rate is slow at the beginning when they have few chunks to exchange with others. Similarly, it can be slow at the end due to the inability of the peer in finding neighbors possessing the few missing chunks that it requires to complete the download. So the authors proposed a simple method to prioritize the use of seeding capacity to certain portions of the file downloading process. They used two ways to choose neighbors to unchoke: (1) Sort-based: a seed sorts its neighbors according to the number of chunks each possesses. Then it unchokes N of them based on the sorting order; (2) Threshold-based: a seed unchokes N neighbors with $[0..K/2]\%$ or $[(100-K/2)..100]\%$ of the chunks. Their experiments use BTSim [9] with the following modifications: (1) nodes could be seeds with different seeding approach; (2) nodes act as free-riders; (3) continuous node arrivals. Their experimental results show that the new seed capacity prioritization approach not only discourages free-riding behavior, but also improves the performance of contributing leechers.

3) Improving Fairness: Bharambe et al. [29] found that BitTorrent's optimistic unchoke mechanism can result in systematic unfairness where a high-capacity node might end up uploading a larger amount of data than it downloads. They proposed two strategies to reduce this unfairness. The first one named Quick bandwidth estimation (QBE) tries to circumvent the need for optimistic unchokes. This is feasible if the nodes can quickly estimate the uploading bandwidths for all its neighboring peers. QBE can estimate a neighbor's upload bandwidth based on the history of past interactions with it. Alternately, the packet-pair principle can be used to estimate the bandwidth. The second proposed algorithm is pairwise block-level TFT, which focuses on fairness of the number of blocks transferred instead of uploading rates. Node A allows its neighbor B to download a block from it if and only if $U_{ab} \leq D_{ab} + \Delta$, where U_{ab} and D_{ab} are the uploaded and the downloaded blocks from node A to/from node B respectively, and Δ is the unfairness threshold for this connection. This ensures that the maximum number of extra blocks served by a node is bounded by $d\Delta$ where d is the size of its neighborhood. The authors used their simulator to evaluate the performance of the new techniques based on three metrics: mean uploading utilization, unfairness, and mean downloading time. The results showed that the proposed methods, QBE and pairwise block-level TFT can effectively address the unfairness towards high-capacity nodes, and also improve the upload link utilization. We should note that QBE is somewhat idealistic since the reliable bandwidth estimation is far from a trivial exercise [64]. The block-level TFT may result in a reduction of uploading utilization because peers can potentially cease to upload when the block-level TFT constraint is not satisfied.

Thommes et al. [64] simulated a simplified version of BitTorrent protocol using Matlab in order to understand the fairness properties. Based on their evaluation, they proposed three methods to improve BitTorrent fairness. The three new mechanisms are as follows: The first method is conditional optimistic unchoke. In this method a peer performs an optimistic unchoke only when it's IFR is larger than 1. The second method is multiple connection chokes, which allows peers to choke/unchoke not just one connection, but multiple connections in each round. A peer computes the Connection Fairness, which is the ratio of the peer's uploading rate to a specific connected peer to the downloading rate from that peer, for each of the five peers to which it is connected. They evaluate two parameters: (a) Threshold Ratio, which is the largest value the Connection Fairness can assume before the corresponding upload is choked; (b) Maximum Chokes (MC), which is the largest number of uploads a peer can choke in each round. If the number of connection fairness values exceeding the Threshold Ratio is larger than the Maximum Chokes, the peer will choke the connections which are unfair. Otherwise, it will choke only MC of the peers. The third mechanism is variable number of outgoing connections, in which the number of connections a peer maintains is variable, instead of a fixed number. The number of outgoing connections varies depending on the peer's upload capacity. They also allow the peer's upload capacity dedicated to each connection to be variable. Each peer evaluates its set of outgoing and incoming connections every 10 seconds. In each iteration, the peer constructs a list L_{nd} of peers to which it is currently uploading, but from which the peer is not receiving any data. The peer will choke the peers in list L_{nd}. Then, the peer constructs another list L_{nu} of peers from which it is downloading, but to which it is not uploading. When $|L_{nu}|$ is larger than $|L_{nd}|$, it starts uploading to a random set of $|L_{nd}|$ peers in L_{nu}. When $|L_{nu}|$ is smaller than $|L_{nd}|$, it begins to upload to all peers in L_{nu}, and randomly choose $|L_{nd}| - |L_{nu}|$ peers to unchoke optimistically.

They defined the Instantaneous Fairness Ratio (IFR) for each peer as the ratio of data uploaded to data downloaded during the previous 10 seconds. If a peer's IFR is less than 1, it means the peer is downloading an excessive amount. Otherwise, it tells us that the peer is downloading an insufficient amount. They evaluated the three modifications by simulation. The results show that the three modifications provide some level of improvement in BitTorrent's fairness.

C. Summary

Table VI summarizes the improvements to BitTorrent's mechanisms that are presented earlier in this section. As can be seen from the table, each suggested improvement addresses one of the issues with BitTorrent mechanisms. No studies of how these improvements interact with each other if included together in BitTorrent are available. It would be interesting to explore the combined effects of the different improvements suggested to see if they indeed work collaboratively to further improve performance, or impede each other.

V. Conclusion

This paper presented a survey of BitTorrent performance. First we reviewed the performance studies of the original BitTorrent's protocols including measurement, simulation, and analytical modeling. We then summarized the findings of these studies. Next, we summarized some of the suggested improvements to BitTorrent's mechanisms in order to further improve its performance.

We note that the innovative mechanisms introduced by BitTorrent has influenced not only P2P file sharing applications, but also served as the foundation for several other applications including P2P streaming and P2P Voice over IP. For example, BitTorrent based streaming applications are described in [65], [66]. Thus we see the study of BitTorrent and its performance as a very useful step not only to understand its performance, but also as a means of learning from the experience to enable better design of similar protocols for other applications.

Acknowledgment

The work described in this paper has been supported by HK RGC under RGC-Competitive Earmarked Research Grant HKUST 617907. The authors would like to thank all the anonymous reviewers for their invaluable suggestions.

References

[1] "Bittorrent," http://www.bittorrent.com/.

[2] B. Cohen, "Incentives build robustness in BitTorrent," in *First Workshop on Economics of Peer-to-peer Systems*, Berkeley, USA, June 2003.

[3] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *SIGCOMM '04: Proc. 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 367–378.

[4] "Gnutella," http://www.gnutelliums.com/.

[5] "Kazaa," http://www.kazaa.com/.

[6] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The BitTorrent p2p file-sharing system: Measurements and analysis," in *IPTPS'05*, 2005, pp. 205–216.

[7] L. Ellis, "BitTorrent's swarms have a deadly bite on broadband nets," Multichannel news, 2006.

[8] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in *IMC '05: Proc. 5th ACM SIGCOMM conference on Internet Measurement*. Berkeley, CA, USA: USENIX Association, 2005, pp. 6–6.

[9] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," in *ICDCS '06: Proc. 26th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2006, p. 66.

[10] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *IEEE INFOCOM*. Miami, USA: IEEE Computer Society, March 2005, pp. 2235–2245.

[11] D. Levin, R. Sherwood, and B. Bhattacharjee, "Fair file swarming with FOX," in *In Fifth International Workshop on Peer-to-peer Systems (IPTPS 2006)*, 2006.

[12] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: a cooperative bulk data transfer protocol," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 941–951, March 2004.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surveys Tutorials*, vol. 7, no. 2, pp. 72–93, Quarter 2005.

[14] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, 2004.

[15] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys Tutorials*, vol. 9, no. 3, pp. 58–74, Quarter 2007.

[16] D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov, "Peer-to-peer overlays for real-time communication: Security issues and solutions," *IEEE Commun. Surveys Tutorials*, vol. 11, no. 1, pp. 4–12, 2009.

[17] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-peer-based resource discovery in global grids: A tutorial," *IEEE Commun. Surveys Tutorials*, vol. 10, no. 2, pp. 6–33, 2008.

[18] M. Izal, G. Urvoy-keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five months in a torrents lifetime," in *Passive and Active Measurements (PAM), LNCS*, vol. 3014, April 2004, pp. 1–11.

[19] A. Bellissimo, B. N. Levine, and P. Shenoy, "Exploring the use of BitTorrent as the basis for a large trace repository," Technical report, University of Massachusetts Amherst, Dept. of Computer Science, Tech. Rep., June 2004.

[20] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in BitTorrent communities," in *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*. New York, NY, USA: ACM, 2005, pp. 111–115.

[21] A. Rasti and R. Rejaie, "Understanding peer-level performance in BitTorrent: A measurement study," *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pp. 109–114, Aug. 2007.

[22] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *IMC '06: Proc. 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 203–216.

[23] K. andré Skevik, V. Goebel, and T. Plagemann, "Analysis of BitTorrent and its use for the design of a p2p based streaming protocol for a hybrid cdn," Delft University of Technology, Tech. Rep., 2004.

[24] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in BitTorrent systems," in *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2007, pp. 301–312.

[25] C. Dale, J. Liu, J. Peters, and B. Li, "Evolution and enhancement of BitTorrent network topologies," *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 1–10, June 2008.

[26] Y. Lei, L. Yang, Q. Jiang, and C. Wu, "Experimental views on neighbor selection in BitTorrent," *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*, pp. 813–818, Sept. 2007.

[27] "Bittornado website," http://www.bittorrent.com/, 2007.

[28] P. A. Felber and E. W. Biersack, "Self-scaling networks for content distribution," in *Int. Workshop on Self-* Properties in Complex Information Systems*, Bertinoro, Italy, May-June 2004.

[29] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a BitTorrent networks performance mechanisms," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–12, April 2006.

[30] E. Adar, "Drawing crowds and bit welfare," *SIGecom Exch.*, vol. 5, no. 4, pp. 31–40, 2005.

[31] A. Al Hamra, A. Legout, and C. Barakat, "Understanding the properties of the BitTorrent overlay," INRIA, Sophia Antipolis, Tech. Rep., July 2007.

[32] G. Urvoy-Keller and P. Michiardi, "Impact of inner parameters and overlay structure on the performance of BitTorrent," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–6, April 2006.

TABLE VI
SUMMARY OF IMPROVED MECHANISMS SUGGESTED FOR BITTORRENT

	Mechanisms	Suggested Improvements	Features of Improvement	Result	Ref.
Overlay Topology	Random Neighbor Selection	Neighbor selection by proximity	Synthetic network coordinates, Euclidean distance	Decrease the file downloading time and increase network utilization	[49], [50]
		Neighbor selection by content	Select neighbor by the most mutually disjoint content	Maximize content availability from neighbors	[52]
	ISP-unawareness	Biased neighbor selection	Choose most neighbors inside the ISP	Reduce the cross-ISP traffic	[9]
		Cost-aware Selection	Select neighbors according to the cost between peers	Reduce the ISP cost	[55]
Piece Exchange	No Collaborative Mechanism	Collaborative Download	Helpers: peers with idle bandwidth can help other peers downloading	Improve download speed; better resource utilization	[57], [58], [59]
		ATFT	Incentives are built on bandwidth instead of content	Several benefits including for new peers, seeds; better resource utilization	[56]
		Helpers	Helpers download a small fraction of file and act as microseeds	Improved download performance for peers	[59]
	Addressing Free-Riding	New incentive mechanism	Peers upload evenly to all links as much as they can under the condition $u - d <= f.c.$	The new proposed mechanism is more robust to prevent free-riding	[60]
		Addressing the problems of large view exploit	Using peer's IP address to track free-riding behavior	Better ability to identify free-riders	[62]
		Seed capacity utilization	Use the seeding capacity to certain portions of a file downloading process	Discourage free-riders' behavior, Improve the leechers' Performance	[63]
	Improving Fairness	QBE; Pairwise block-level TFT	QBE: uploading bandwidths can be estimated quickly; Pairwise block-level TFT: the maximum number of extra blocks served by a node is bounded	Address the problems of low uplink utilization and unfairness	[29]
		Conditional optimistic unchoke; Multiple connection chokes; Variable number of outgoing connections	Conditional optimistic unchoke: only when a peer's IFR is larger than 1, it performs an optimistic unchoke algorithm; Multiple connection chokes: peers can choke/unchoke multiple connections in each round; Variable number of outgoing connections: the number of connections a peer maintains is variable	Improve fairness	[64]

[33] G. Wu and T. cker Chiueh, "How efficient is BitTorrent?" in *Multimedia Computing and Networking 2006*, S. Chandra and C. Griwodz, Eds., vol. 6071, no. 1. SPIE, 2006, p. 60710O. [Online]. Available: http://link.aip.org/link/?PSI/6071/60710O/1

[34] A. Bharambe, C. Herley, and V. Padmanabhan, "Microsoft research simulator for the BitTorrent protocol," http://research.microsoft.com/projects/btsim/, 2006.

[35] X. Yang and G. de Veciana, "Performance of peer-to-peer networks: Service capacity and role of resource sharing policies," *Performance Evaluation*, vol. 63, no. 3, pp. 175 – 194, 2006, p2P Computing Systems. [Online]. Available: http://www.sciencedirect.com/science/article/B6V13-4FJGW2F-1/2/be7c36ff2970dd2cea798f2de6200a40

[36] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 155–169, Jan. 2007.

[37] D. Arthur and R. Panigraphy, "Analyzing the efficiency of BitTorrent and related peer-to-peer networks," in *Proc. Seventeenth annual ACM-SIAM symposium on Discrete algorithms*, January 2005, pp. 961–969.

[38] R. Kumar and K. Ross, "Peer-assisted file distribution: The minimum distribution time," *Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on*, pp. 1–11, Nov. 2006.

[39] J. Yu, M. Li, F. Hong, and G. Xue, "Free-riding analysis of BitTorrent-like peer-to-peer networks," *Services Computing, 2006. APSCC '06. IEEE Asia-Pacific Conference on*, pp. 534–538, Dec. 2006.

[40] Y. Tian, D. Wu, and K. W. Ng, "Modeling, analysis and improvement for BitTorrent-like file sharing networks," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–11, April 2006.

[41] P. Saraswat and P. Batra, "An empirical performance evaluation and modelling of BitTorrent peer-to-peer file sharing system using queuing network models," Research project, Advanced Learning and Research Institute, Switzerland, 2007.

[42] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in BitTorrent?" in *NSDI'07*, Cambridge, MA, April 2007, pp. 1–14.

[43] F. Lo Piccolo and G. Neglia, "The effect of heterogeneous link capacities in BitTorrent-like file sharing systems," *Peer-to-Peer Systems, 2004. International Workshop on Hot Topics in*, pp. 40–47, Oct. 2004.

[44] F. Clévenot-Perronnin, P. Nain, and K. W. Ross, "Multiclass p2p networks: Static resource allocation for service differentiation and bandwidth diversity," *Performance Evaluation*, vol. 62, no. 1-4, pp. 32 – 49, 2005, performance 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6V13-4GV9S5V-5/2/182bf23d02fcbeff01e45a97ed891a9a

[45] B. Fan, D.-M. Chiu, and J. Lui, "Stochastic analysis and file availability

enhancement for BT-like file sharing systems," *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pp. 30–39, June 2006.
[46] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *Journal of Scheduling*, vol. 11, no. 2, pp. 105–120, April 2008.
[47] W.-C. Liao, F. Papadopoulos, and K. Psounis, "Performance analysis of BitTorrent-like systems with heterogeneous users," *Performance Evaluation*, vol. 64, no. 9-12, pp. 876 – 891, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6V13-4P192CX-2/2/d13c7120e07234e69160c85b5b861f8d
[48] M. Bertoli, G. Casale, and G. Serazzi, "An overview of the jmt queuing network simulator," Politecnico di Milano, Tech. Rep., 2007.
[49] A. Qureshi, "Exploring proximity based peer selection in a BitTorrent-like protocol," MIT 6.824 student project, 2004.
[50] L. Zhang, J. K. Muppala, and W. Tu, "Exploiting proximity in cooperative download of large files in peer-to-peer networks," *International Conference on Internet and Web Applications and Services, (ICIW'07)*, vol. 0, p. 1, 2007.
[51] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris, "Lighthouses for scalable distributed location," in *Second International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Feb 2003, pp. 278–291.
[52] S. G. Koo, K. Kannan, and C. G. Lee, "On neighbor-selection strategy in hybrid peer-to-peer networks," *Future Generation Computer Systems*, vol. 22, no. 7, pp. 732 – 741, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6V06-4JRKCPC-2/2/e800f30606343b0a99ebf78c36033a68
[53] "Ns-2, the network simulator," http://www.isi.edu/nsnam/ns/, 2003.
[54] V. Aggarwal, A. Feldmann, C. Scheideler, and M. Faloutsos, "Can isps and p2p users cooperate for improved performance," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 29–40, 2007.
[55] S. Yamazaki, H. Tode, and K. Murakami, "CAT: A cost-aware BitTorrent," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, Oct 2007, pp. 226–227.
[56] P. Garbacki, D. Epema, and M. van Steen, "An amortized tit-for-tat protocol for exchanging bandwidth instead of content in p2p networks," *Self-Adaptive and Self-Organizing Systems, 2007. SASO '07. First International Conference on*, pp. 119–128, July 2007.
[57] P. Garbacki, A. Iosup, D. Epema, and M. van Steen, "2fast: Collaborative downloads in p2p networks," in *P2P '06: Proc. Sixth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 23–30.
[58] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips, "Tribler: A social-based peer-to-peer system," *Concurrency and Computation: Practice & Experience*, vol. 20, no. 2, pp. 127–138, Feb 2008.
[59] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the role of helpers in peer-to-peer file download systems: Design, analysis and simulation," in *IPTPS'07*, 2007.
[60] S. Jun and M. Ahamad, "Incentives in BitTorrent induce free riding," in *P2PECON '05: Proc. 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*. New York, NY, USA: ACM, 2005, pp. 116–121.
[61] "Planetlab," http://www.planet-lab.org/.
[62] M. Sirivianos, J. Park, R. Chen, and X. Yang, "Freeriding in BitTorrent networks with the large view exploit," in *IPTPS'07*, 2007.
[63] A. L. Chow, L. Golubchik, and V. Misra, "Improving BitTorrent: A simple approach," in *Proceedings of IPTPS*, 2008.
[64] R. Thommes and M. Coates, "BitTorrent fairness: analysis and improvements," in *Workshop Internet, Telecom. and Signal Proc. (WITSP'05)*, Dec 2005.
[65] S. Tewari and L. Kleinrock, "Analytical model for BitTorrent-based live video streaming," *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pp. 976–980, Jan. 2007.
[66] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–6, April 2006.

Raymond Lei. Xia is a M. Phil. student in The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.

Jogesh K. Muppala (M'85-SM'02) received the Ph. D. degree in Electrical Engineering from Duke University, Durham, NC in 1991. He is currently an associate professor in the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. He was previously a Member of the Technical Staff at Software Productivity Consortium (Herndon, Virginia, USA) from 1991 to 1992, where he was involved in the development of modeling techniques for systems and software. While at Duke University, he participated in the development of two modeling tools, the Stochastic Petri Net Package (SPNP) and the symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE), both of which are being used in several universities and industry in the USA. He was the program co-chair for the 1999 Pacific Rim International Symposium on Dependable Computing held in Hong Kong in December 1999. He also co-founded and organized The Asia-Pacific Workshop on Embedded System Education and Research. He has also served on program committees of many international conferences. He received the Excellence in Teaching Innovation Award 2007. He was also awarded the Teaching Excellence Appreciation Award by the Dean of Engineering, HKUST. Dr. Muppala is a Senior Member of IEEE, IEEE Computer Society and IEEE Communications Society, and a participating representative from HKUST with EDUCAUSE.