BitTorrent: An Extensible Heterogeneous Model

Alix L.H. Chow
University of Southern California

Leana Golubchik
University of Southern California

Vishal Misra
Columbia University

***Abstract*—Peer-to-peer (P2P) systems in general, and BitTorrent (BT) specifically, have been of significant interest to researchers and Internet users alike. Existing models of BT abstract away certain characteristics of the protocol that are important, which we address in this work. We present a simple yet accurate and easily extensible model of BT. The model's accuracy is validated through a rigorous simulation-based study and its extensibility is illustrated by incorporating recently proposed approaches to protocol changes in BT.**

I. Introduction

BitTorrent (BT) [1] has been of significant interest to researchers and Internet users alike. In this work, we focus on a simple model of BT, as a tool for providing insight into its behavior and possible future improvements in its architecture.

In BT, nodes join the system and begin exchanging data chunks with their neighbors. Nodes that do not have a complete copy of the file are termed "leechers" and those that do are termed "seeds". Nodes that do not contribute their upload capacities are termed "free-riders". Each leecher i picks ("unchokes") a number of nodes to whose requests it will respond with an upload of an appropriate chunk. A subset of these nodes is picked based on the tit-for-tat (TFT) mechanism, i.e., those neighbors that have provided the best service (in terms of download rate) to node i recently. Another subset is picked randomly, i.e., they are "optimistically unchoked" (to explore better neighbors). Seeds also pick a subset of neighbors, and upload data to them. In a more recent version of BT ([2]) the seeding capacity is distributed essentially uniformly to the neighboring peers. All these choices are re-evaluated periodically.

A number of works have focused on modeling BT or BT-like systems (see Section V) - some model flash crowd behavior and some steady state behavior. In this work, we focus on steady state behavior. Within the class of efforts that model steady state behavior, *our model is distinct as it accounts for: (1) seeds and (2) free-riders*. Moreover, to improve the accuracy of our model, we account for: (a) imperfect clustering behavior in regular (TFT-based) unchokes and (b) bias in optimistic unchokes.

We also note that our model is of a *heterogeneous* BT system (in terms of node bandwidths) and that the only other work that considers a heterogeneous system in the context of steady state behavior is [3], but without the distinctions stated above. We first explain why it is important to include seeding and free-riding behavior in a BT model; we then present our model in Section II and illustrate its accuracy in Section III (where we also compare our model to the only other available model of a heterogeneous BT system in steady state, given in [3]).

A number of research studies (e.g., [3], [4], [5], [6]), have focused on the fairness, robustness, and performance characteristics of BT, mainly resulting from the TFT mechanism. However, very few studies (except [7]) have considered seeding behavior effects. Given typical user behavior and the design of most BT clients, seeding is a common phenomenon seen in the BT ecosystem. Many users leave their BT clients unmonitored while performing a large download, and the clients seamlessly transition to being seeds once the download completes. Also, some sharing communities enforce a download/upload ratio to enable seeding and a better performing system. Real-world measurements (e.g., [2], [8], [9], [10]) suggest that, in most torrents, there exists a significant number of seeds, with a large variety of seeding capacity. Some of the torrents measured have twice the number of seeds than the number of leechers.

Seeding contributes capacity to the system, which can compensate for the asymmetric bandwidth scenarios in the Internet. However, it can degrade the fairness and incentive properties of the system, as free-riders can finish their downloads with reasonable performance by relying only on the seeds - not only do they not contribute to the systems' upload capacity, they also effectively reduce the performance gains provided by seeds.

One important thread in the design and evaluation of P2P systems has been the provision of incentives for nodes to contribute their resources. An interesting aspect of BT, which makes it stand out in that respect, is the "forcing" of peers to share their resources *while* attempting to complete their downloads (through the TFT mechanism) – this is done "locally" and without a need for centralized or distributed mechanisms for implementing an incentive system. Although this works nicely, there are still opportunities for free-riders to download files using: (1) capacity provided by leechers through the optimistic unchoking mechanism (used by leechers to probe for new downloading opportunities), and (2) capacity provided by seeds.

Since (a) there are significant seeding resources observed in typical torrents, which have significant effects on BT performance, and (b) there are significant opportunities for free-riders to hurt BT performance, we believe that it is important to include such behavior in an accurate BT model. Thus, in our work, we provide a simple and extensible model that can characterize the effect of seeding and free-riding in BT.

Another aspect that we include in our model is that of imperfect clustering[1]. Previous approaches have assumed that

This research was funded in part by the USC Annenberg Graduate Fellowship and the NSF 0627590, 0615126, 0720612, 0540420 grants.

[1]Our notion of clustering is similar to that in [11].

operationally BT decomposes cleanly into clusters, where nodes "find" like nodes that are similar in bandwidth capacity and exchange chunks amongst each other. That however is not the case in practice, and we model that, with details given in Section II. Specifically, the contributions of our work are as follows:

- We provide a simple and complete steady state model of a heterogeneous BT system. To the best of our knowledge, it is the first analytical performance model that includes the behavior of seeds and free-riders. Our model predictions confirm that BT is indeed quite exploitable.
- We extend it to include important BT characteristics, namely (i) imperfect clustering in regular (TFT-based) unchokes and (ii) bias in optimistic unchokes.
- We validate our model using simulation results and demonstrate the importance of including imperfect clustering and biased optimistic unchoking in having accurate prediction of nodes' download time in BT.
- Our model's extensibility is demonstrated by explicitly modeling two variations of BT, (a) the well known large view exploit [12], [13], and (b) a recently proposed fix for the large view exploit [7].

II. BT Model

We use a simple rate balance model, similar to [3], but we account for a number of BT characteristics (which are not included in [3]). Let h be the number of node classes, where each class is defined by its upload and download capacities, U^i and D^i, respectively. New nodes arrive to the system at an average rate of λ, and there is a probability of p^i that a newly arrived node belongs to class i - thus, class i nodes arrive at a rate of $\lambda^i = p^i\lambda$. A class i node downloads at the rate of d^i chunks per time unit ($d^i \leq D^i$), and it uploads at the rate of u^i chunks per time unit. Since (a) Internet upload and download capacities are often asymmetric and (b) some users tend to limit their upload capacities in BT, we assume that the upload links are the bottlenecks, i.e., $u^i = U^i$. This is a fairly typical assumption in the literature, e.g., it is also made in [3].

In steady state, there are N_l^i class i leechers in the system and N_s^i class i seeds, where $N^i = N_l^i + N_s^i$ represents the total number of class i nodes. Let T_l^i be the average amount of time a class i leecher takes to download a file. Let m be the number of data chunks in the file; then, $d^i = \frac{m}{T_l^i}$. Let T_s^i be the average amount of time a class i node stays in the system after becoming a seed. Thus, $T^i = T_l^i + T_s^i$ is the average amount of time a class i node stays in the system. Then, we can state that at any time, the total download rate of all leechers is equal to the total upload rate from all leechers and all seeds, i.e.:

$$\sum_{i=1}^{h} u^i N^i = \sum_{i=1}^{h} d^i N_l^i.$$

Next, we build on this fairly coarse model while making some standard assumptions, including that the system is in steady state and that it has a sufficiently large number of nodes where each peer has a sufficiently large number of neighbors[2]. Since in BT a node receives its downloads from other leechers (via regular and optimistic unchokes) and from seeds, what is needed is a model of how much download capacity a node will receive due to each. That is what we proceed to derive next. We do this first by assuming that there is (a) *perfect clustering* among nodes of the same class and (b) that the optimistic unchokes are unbiased. (This also corresponds to the assumptions made in [3].) We then show how to relax these assumptions in order to obtain a more accurate model.

Regular Unchokes: We first assume perfect clustering, i.e., that exchange of chunks due to TFT only occurs between nodes of the same class. Let x be the number of simultaneous unchokes performed by a node, where x_r of these correspond to regular unchokes (due to TFT) and the remaining x_o correspond to optimistic unchokes. Since (given perfect clustering) the upload capacity due to regular unchokes of a class i node is distributed to nodes of class i *only*, we describe the download rate of a class i node due to regular unchokes (from leechers of the same class), d_{reg}^i, as:

$$d_{reg}^i = \frac{N_l^i u^i \frac{x_r}{x}}{N_l^i} = u^i \frac{x_r}{x}.$$

Optimistic Unchokes: Here, we assume optimistic unchokes are unbiased, i.e., that they are uniformly distributed among all leechers in the system. Thus, we describe the download rate that a class i node due to optimistic unchokes (from all other leechers), d_{opt}^i, as:

$$d_{opt}^i = \frac{\sum_{j=1}^{h} N_l^j u^j \frac{x_o}{x}}{N_l}.$$

Seed Unchokes: We assume seeds upload uniformly to all leechers (approximating the newer BT protocol) and describe the download rate of a class i node from all seeds, d_{seed}^i, as:

$$d_{seed}^i = \frac{\sum_{j=1}^{h} N_s^j u^j}{N_l}.$$

Given above, we can state the following for the class i node's download rate, $d^i = d_{reg}^i + d_{opt}^i + d_{seed}^i$:

$$d^i = u^i \frac{x_r}{x} + \frac{\sum_{j=1}^{h} N_l^j u^j \frac{x_o}{x}}{N_l} + \frac{\sum_{j=1}^{h} N_s^j u^j}{N_l}. \quad (1)$$

Next, we apply Little's Result to obtain:

$$N^i = \lambda^i T^i, N_l^i = \lambda^i T_l^i,$$
$$N_s^i = N^i - N_l^i = \lambda^i (T^i - T_l^i) = \lambda^i T_s^i.$$

Since, $T_l^i = \frac{m}{d^i}$, we can plug this, along with $\lambda^i = p^i\lambda$, into Equation (1), to obtain:

$$\begin{aligned} d^i &= u^i \frac{x_r}{x} + \frac{\sum_{j=1}^{h} p^j \lambda \frac{m}{d^j} u^j \frac{x_o}{x}}{\sum_{j=1}^{h} p^j \lambda \frac{m}{d^j}} + \frac{\sum_{j=1}^{h} p^j \lambda T_s^j u^j}{\sum_{j=1}^{h} p^j \lambda \frac{m}{d^j}} \\ &= u^i \frac{x_r}{x} + \frac{\sum_{j=1}^{h} \frac{p^j u^j x_o}{d^j x}}{\sum_{j=1}^{h} \frac{p^j}{d^j}} + \frac{\sum_{j=1}^{h} p^j T_s^j u^j}{\sum_{j=1}^{h} \frac{p^j m}{d^j}}. \quad (2) \end{aligned}$$

[2]We also do not model the initial slow startup of newly arrived nodes, and assume that leechers do not abort in the middle of a download.

Thus, we have a set of h equations with h unknowns, $d^1, d^2, ..., d^h$, which can be solved to obtain the average download rate of each class. Note that the above model degenerates to the one presented in [3], if we ignore the seeds. However, as noted in Section I, seeds have a significant effect on BT systems and thus need to be modeled. As also noted in Section I, free-riders have a significant effect as well; thus, we address inclusion of free-riders in the model next.

A. Free-Riders

A simple approach to including free-riders in this model is to view them as another user class (or multiple classes, if they have different download capacities), all with upload capacity of 0. Then, the download rate of a free-riding class is described as that of a contributing leecher class, but with the first term (corresponding to TFT) in Equation (2) dropped; e.g., if we had one free-riding class, say class h, then

$$d^h = \frac{\sum_{j=1}^{h} \frac{p^j u^j x_o}{d^j x}}{\sum_{j=1}^{h} \frac{p^j}{d^j}} + \frac{\sum_{j=1}^{h} p^j T_s^j u^j}{\sum_{j=1}^{h} \frac{p^j m}{d^j}},$$

would be its download rate, where classes 1 through $h-1$ correspond to contributing leechers - their download rate equations remain as in Equation (2), with $u^h = 0$. This, of course, can be done for multiple free-riding classes.

We believe that the basic model presented above is quite adaptable; thus future architectural and protocol changes can be incorporated and studied through it (refer to Section IV). However, we first refine our model to make it more accurate.

B. More Realistic Model

The above model assumes that regular unchokes are perfectly clustered and that optimistic unchokes are unbiased (i.e., as in [3]). However, evidence based on measurements of real BT systems in [11] and our simulations in Section III clearly indicates that significant imperfect clustering (in regular unchokes) exists. Thus, here we remove these assumptions to more accurately account for how the real BT protocol works. Specifically, we first modify the model in a more abstract manner, and then further develop it based on potential causes of imperfect clustering and biased optimistic unchokes.

Firstly, we take into account the fact that a fraction of regular unchokes will go to nodes in other classes. Specifically, we define $q_{i,j}$ to be the fraction of regular unchokes from class i that will go to class j, with $\sum_{j=1}^{h} q_{i,j} = 1$. Given $q_{i,j}$, the download rate that a class i node receives from regular unchokes from all nodes, $d_{reg}^{i'}$, can be described as:

$$d_{reg}^{i'} = \frac{\sum_{j=1}^{h} q_{j,i} N_l^j u^j \frac{x_r}{x}}{N_l^i}.$$

Secondly, we consider the fact that optimistic unchokes of a node are not distributed evenly to all leechers. Specifically, we define $o_{i,j}$ to be the fraction of optimistic unchokes from class i that will go to class j, where $\sum_{j=1}^{h} o_{i,j} = 1$. Given $o_{i,j}$, the download rate a class i node receives from optimistic unchokes from all nodes, $d_{opt}^{i'}$, can be described as:

$$d_{opt}^{i'} = \frac{\sum_{j=1}^{h} o_{j,i} N_l^j u^j \frac{x_o}{x}}{N_l^i}.$$

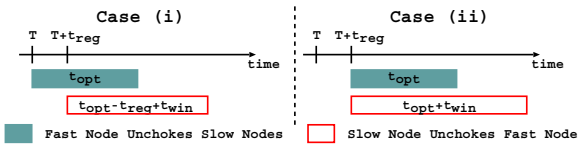

Fig. 1. Slow Node Imperfect Clustering

Given these adjustments to regular and optimistic unchokes, we can express our new model for the download rate of a class i node, $d^{i'} = d_{reg}^{i'} + d_{opt}^{i'} + d_{seed}^{i}$, as:

$$\begin{aligned} d^{i'} &= \frac{\sum_{j=1}^{h} q_{j,i} N_l^j u^j \frac{x_r}{x}}{N_l^i} + \frac{\sum_{j=1}^{h} o_{j,i} N_l^j u^j \frac{x_o}{x}}{N_l^i} \\ &\quad + \frac{\sum_{j=1}^{h} N_s^j u^j}{N_l} \\ &= \frac{d^{i'}}{p^i} \sum_{j=1}^{h} \frac{(q_{j,i} x_r + o_{j,i} x_o) p^j u^j}{d^{j'} x} + \frac{\sum_{j=1}^{h} p^j T_s^j u^j}{\sum_{j=1}^{h} \frac{p^j m}{d^{j'}}}, \end{aligned} \quad (3)$$

where the main challenge is in determining $q_{i,j}$ and $o_{i,j}$. To do this, we examine possible causes for imperfect clustering and biased optimistic unchokes.

Imperfect Clustering: Although there are a number of potential causes, we conjecture that the main cause of imperfect clustering is that optimistic unchokes from faster classes of nodes destroy clustering of slower classes of nodes. Specifically, when a fast node optimistically unchokes a slow node, the slow node reciprocates with a regular unchoke (as it is getting a high download rate from that fast node).

For ease of exposition, we first focus on a two class ($slow$ and $fast$) scenario. We assume that the fast nodes are perfectly clustered ($q_{fast,fast} = 1$ and $q_{fast,slow} = 0$). We then derive the effects of imperfect clustering on slow nodes by considering the timing of regular and optimistic unchokes between slow and fast nodes[3]. Specifically, the reciprocation of slow nodes with regular unchokes to optimistic unchokes of fast nodes is *temporary*. As a fast node realizes that it unchoked a slow node (due to a relatively slow download rate from that node), it discards the slow node. Since the TFT mechanism re-evaluates choices of best download rates using a sliding window, the effect of the fast node's optimistic unchoke eventually wears off, at which point the slow node stops the regular unchoke of the fast node. The extreme cases of how this can happen are illustrated in Figure 1[4]. Here we define t_{opt} as the optimistic unchoking re-evaluation interval, t_{reg} as the regular unchoking re-evaluation interval, and t_{win} as the history window size for

[3]This is similar to the derivation in [4], which was done in the context of a flash crowd, rather than a steady state, model.

[4]A similar method can be used to model the effect of fast nodes doing regular unchokes to slow nodes. However, since measurement and simulation evidence indicates that this is a less frequent case, we omit it here as it would further complicate the model.

re-evaluation of both unchoking mechanisms. We now examine the two cases in Figure 1. If the optimistic unchoke arrives right after the regular unchoke re-evaluation period of a slow node, say T (as depicted by the shaded box in Case (i)), then the slow node reciprocates with a regular unchoke to the fast node, starting in the next re-evaluation interval, $T + t_{reg}$, for a time interval of $t_{opt} - t_{reg} + t_{win}$, as depicted by the clear box in the figure. At the other extreme, if the unchoke arrives right before the next re-evaluation interval, i.e., Case (ii), the slow node reciprocates with a regular unchoke to the fast node, for a time interval of $t_{opt} + t_{win}$. Here, we assume that the unchokes arrive uniformly during the regular unchoking re-evaluation interval. As the slow node does x_r regular unchokes at the same time, on average, the fraction of the regular unchoking capacity of a slow node that is spent on a fast node due to a fast node's optimistic unchoke, f_{opt}, can be described as:

$$f_{opt} = \frac{\frac{1}{2}\big((t_{opt} - t_{reg} + t_{win})) + (t_{opt} + t_{win})\big)}{x_r t_{opt}}.$$

We can then approximate $q_{slow,fast}$, as

$$q_{slow,fast} = min(g_{opt} f_{opt}, 1), \tag{4}$$

where g_{opt} is the average number of optimistic unchokes that a slow node is receiving from fast nodes[5]. Note that, $q_{slow,slow} = 1 - q_{slow,fast}$. We can approximate g_{opt} by looking at the average number of fast and slow peers of a node. We define s as the peer set size of a node (i.e., the number of peers to which that node connects and exchanges data with), and we approximate g_{opt} by:

$$g_{opt} = s\frac{N_l^{fast}}{N_l} x_o o_{fast,slow} \frac{1}{s\frac{N_l^{slow}}{N_l}} = \frac{N_l^{fast}}{N_l^{slow}} x_o o_{fast,slow},$$

where $o_{fast,slow}$ is the fraction of optimistic unchokes from the fast class that goes to the slow class (as discussed in more detail below). Here, $s\frac{N_l^{fast}}{N_l}$ and $s\frac{N_l^{slow}}{N_l}$ is the average number of fast peers and slow peers of a node, respectively.

We now show how to extend this to more than two classes. Without loss of generality, we assume that the class indices are in *descending order* of their uploading capacities. As before, we assume that faster nodes do not experience imperfect clustering with slower nodes, i.e., $q_{i,j} = 0, \forall j > i$. Thus, there are no regular unchokes from a faster node to a slower node. For the regular unchokes of faster nodes by slower nodes, we can use the following approximation:

$$q_{i,j} = min\big(min(g_{i,j} f_{opt}, 1), 1 - \sum_{k=1}^{j-1} q_{i,k}\big), \tag{5}$$

where $q_{i,i} = 1 - \sum_{j=1}^{i-1} q_{i,j}$, $q_{i,1} = min(g_{i,1} f_{opt}, 1)$, and $g_{i,j}$ is the average number of optimistic unchokes that a class i node is receiving from class j nodes, which can be described as:

$$g_{i,j} = s\frac{N_l^j}{N_l} x_o o_{j,i} \frac{1}{s\frac{N_l^i}{N_l}} = \frac{N_l^j}{N_l^i} x_o o_{j,i}.$$

The idea is that when we consider the regular unchoking capacity due to imperfect clustering of class i, we assume that as much as possible of it first goes to the fastest class in the system, i.e., $min(g_{i,1} f_{opt}, 1)$. Then, as much as possible of the remainder goes to the second fastest class, and so on.

Biased Optimistic Unchoking: Although usually modeled as being uniform among all leechers (as in [3]), optimistic unchokes are in fact biased in the real world[6], with the reason being that optimistic unchokes are only performed on peers that are not (currently) unchoked through regular unchokes. Thus, given $q_{i,j}$, we can approximate $o_{i,j}$ as:

$$o_{i,j} = \frac{s\frac{N_l^j}{N_l} - x_r q_{i,j}}{\sum_{k=1}^{h}(s\frac{N_l^k}{N_l} - x_r q_{i,k})}. \tag{6}$$

Note that, the more perfect is the clustering (due to regular unchokes), the more biased are the optimistic unchokes. However, the biasing effect becomes less significant when the peer set size, s, becomes larger. Here we assume that s is large enough such that $s\frac{N_l^j}{N_l} > x_r q_{i,j}$ for all i, j.

The model can now be solved numerically, e.g., using fixed point iteration.

III. Model Validation and Insight

In this section, we validate the model proposed in Section II using simulation. (Validation of the model using real-world experiments is an ongoing effort.) We also illustrate how our model can be used to obtain insight into the BT system. We use the BT simulator provided by [14] (also used by other researchers in the community), with modifications described below. The simulator is event-based and simulates the chunk exchanging mechanism of the BT protocol.

We extended the simulator to support: (a) nodes staying around as seeds, (b) node arrivals, and (c) nodes acting as free-riders (i.e., nodes that do not unchoke and leave the system upon download completion). The seeding times and node inter-arrival times follow an exponential distribution. We also modified the original seeding scheme to be more uniform, in-line with the current BT protocol. Moreover, we also fixed a bug in the original simulator that affected the selection of peers for unchoking - the original simulator incorrectly implemented that part of the BT protocol, which resulted in a higher probability of unchoking previously unchoked peers.

Unless specified otherwise, the following results correspond to the simulation settings in Table I. The system starts with 1 origin seed with a 1000 kbps upload capacity, that stays in the system for 12 hours. Nodes arrive to the system from a Poisson process with a rate λ and are assigned to a particular class according to a given distribution. The classes differ in their upload and download capacities. We consider the steady

[5] In cases where $g_{opt} f_{opt} > 1$, all the regular unchoking capacity of a slow node is spent on g_{opt} fast nodes. Although effectively this means that the fraction of regular unchoking capacity spent on each fast node would be less than f_{opt}, this will not affect our results as in our model we are only interested in $q_{slow,fast}$.

[6] We also observe this in our simulations.

TABLE I
SETTINGS

Filesize (m)	500 MB (2000 Chunks, 256 KB each)
Avg node inter-arrival ($\frac{1}{\lambda}$)	1 min
Peer Set Size (s)	80
# Leecher Unchokes	4 Reg. (x_t) + 2 Opt. (x_o)
# Seed Unchokes	6
Unchoke Re-eval. Interval	Reg. (t_{reg}): 5 sec; Opt. (t_{opt}): 30 sec
Re-eval. History (t_{win})	20 sec

TABLE II
CLASS DESCRIPTION (TWO CONTRIBUTING CLASSES)

Class	Download Capacity	Upload Capacity
Fast	5000kbps	512kbps
Slow	5000kbps	128kbps

state behavior of the system. Each simulation run corresponds to 63 hours, where we only compute our results over the last 48 hours. (We check our results to make sure the system passes the ramp up stage during the first 15 hours.)

The model-based results are obtained numerically, using a fixed point iteration method. These solutions converge quickly, even for multiple class cases.

We note that a single simulation run (i.e., one point in a figure) takes more than 10 hours on a reasonable Core 2 Duo machine while our model can compute the entire figure in less than 1 sec. Thus, our model provides a much faster way (than simulation) of exploring system parameters and design choices.

In what follows, "Model-PC" refers to the basic model, i.e., described by Equation 2, and "Model-IC" refers to our final model with imperfect clustering and biased optimistic unchoking enhancements, i.e., described by Equation 3.

Experiment 1: Two Leecher Classes. We consider a system with two classes and bandwidth settings given in Table II and no seeding time. Figure 2 depicts the resulting download times as a function of the percentage of node arrivals corresponding to the fast class, where we observe the following. Model-IC is much more accurate in predicting the download time of the slow class. For example, when 80% of the arriving nodes are from the fast class, Model-PC's prediction for the slow class download time differs from simulation by $\approx 22\%$ while Model-IC's prediction differs by less than 1%. The error in Model-PC is mainly due to neglecting imperfect clustering. This can be observed from there being a larger error with a higher percentage of fast nodes and from Figure 3 which depicts the fraction of the regular unchoking capacity that is not distributed to nodes of the same class, obtained from the simulation and from Model-IC. As can be seen, slow nodes have a higher fraction of imperfect clustering when there is a higher percentage of fast nodes among the arrivals.

Model-IC can predict the *trend* in the imperfect clustering of slow nodes; however, it under-estimates the number, which can be explained as follows. While in the model we assumed that fast nodes have perfect clustering, simulation results indicate that some degree of imperfect clustering exists among fast nodes as well. Thus, the under-estimation of imperfect clustering of slow nodes is due to the imperfect clustering of fast nodes - the regular unchokes from fast nodes to slow nodes further degrade clustering of slow nodes, similarly to the degradation due to fast nodes' optimistic unchokes of slow nodes. To correct for the phenomenon, we can use a similar derivation to the one given in Section II [7].

TABLE III
CLASS DESCRIPTION (WITH FREE-RIDERS)

Class	Fraction	Download Capacity	Upload Capacity
Contributing	80%	5000kbps	512kbps
Free-Riders	20%	5000kbps	0

Experiment 2: Three Leecher Classes. Next, we look at a system with three classes, where we add a "Super-Fast" class, with upload (download) capacity of $1000kbps$ ($5000kbps$), to the classes given in Table II (again, with no seeding time at first). Figure 4 depicts the average download times of the three classes as a function of the percentage of super-fast node arrivals (with the remainder of arrivals split evenly between the fast and slow classes). Observe that Model-IC predicts the download times quite accurately, while Model-PC results in a larger error - e.g., when 70% of the arrivals are from the super-fast class, Model-PC has an error of $\approx 20\%$ and $\approx 16\%$ for the slow and the fast class, respectively, while the corresponding errors in Model-IC are only $\approx 2\%$ and $\approx 5\%$. This is mainly due to neglecting of imperfect clustering in Model-PC.

To further illustrate the difference between Model-IC and Model-PC, in Figure 5 we depict the average download times of the three classes as a function of the average seeding time, with 70%, 15%, and 15% of the arrivals corresponding to super-fast, fast, and slow classes, respectively. From the results, we observe that Model-IC can predict the download times of all classes quite well. Again, we note that in this figure, Model-PC would degenerate to the model in [3], at 0 seeding time, but since the model in [3] does not include seeding behavior, its prediction would not change as the seeding time is increased.

Experiment 3: Free-Riders. We now focus on free-riders. For clarity of presentation, we look at the case with one contributing leecher class (CL) and one free-riders class (FR), with bandwidth setting given in Table III. Figure 6 depicts the average download times of the contributing leechers and free-riders as a function of the fraction of percentage of free-riders in the arriving nodes, where contributing nodes have an average seeding time of 120 min. We observe the following. Both models can predict the download times of contributing leechers and free-riders quite accurately. The existence of free-riders slows down contributing leechers. Thus, it is important to include free-riders in the model. Moreover, improvements in contributing leechers' download times can be achieved by discouraging free-riders, e.g., if we can reduce the percentage of free-riders from 20% to 5%, we can speed up the downloads of contributing leechers by more than 50%. Thus, it is worth considering and modeling schemes which discourage free-riding, as discussed in Section IV.

Figure 7 depicts the average download times of contributing leechers and free-riders as a function of the contributing nodes'

[7] Although the degree of imperfect clustering is under-estimated, the download time prediction is accurate, e.g., because the loss of regular unchoking capacity within the slow class is compensated by the fast nodes' regular unchoking capacity given to the slow class.

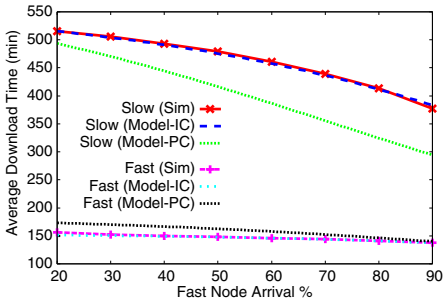

Fig. 2. Different Mix of Two Contributing Classes.

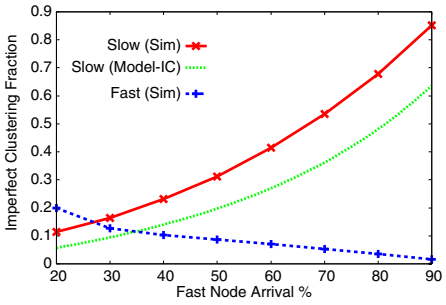

Fig. 3. Imperfect Clustering Fraction.

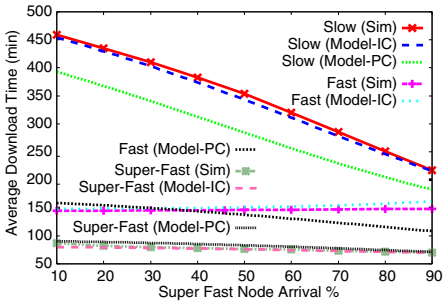

Fig. 4. Different Mix of Three Contributing Classes.

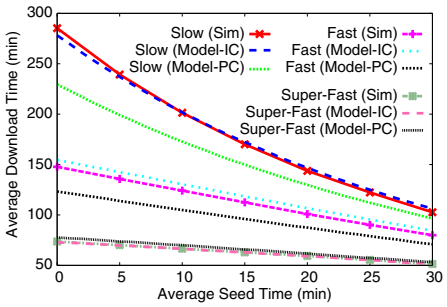

Fig. 5. Different Seed Time (Three Classes).

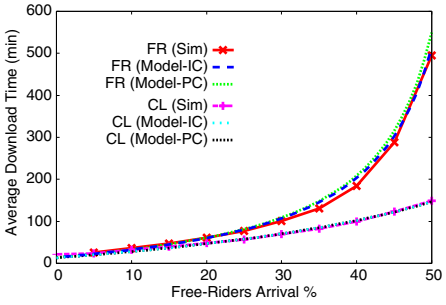

Fig. 6. Different Fraction of Free-Riders.

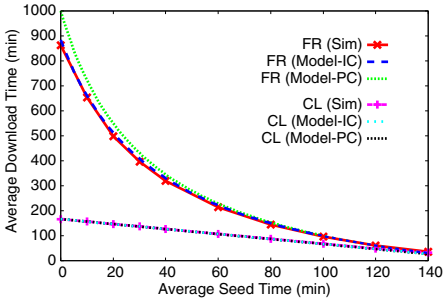

Fig. 7. Different Seed Time (with FR).

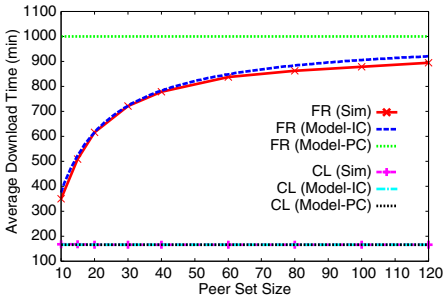

Fig. 8. Different Peer Set Size

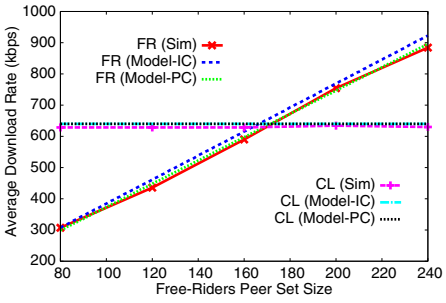

Fig. 9. Large View Exploit.

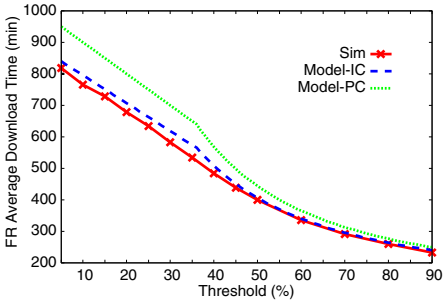

Fig. 10. Different Threshold (Free-Riders).

average seeding time, with bandwidth settings given in Table III. Observe that both models can predict download times quite well when the seeding time is relatively high. Since the seed unchokes are modeled fairly accurately in both models and since the effect of seeding capacities dominates as that capacity grows, Model-PC exhibits similar accuracy to Model-IC at higher seeding capacity. Model-IC is more accurate when the seeding time is shorter, e.g., with no seeding time, Model-PC's prediction of free-riders' download time differs from that of simulation by $\approx 16\%$, whereas as that of Model-IC's only differs by $\approx 2\%$. As there is only one contributing class, the regular unchokes are close to perfect (as also observed in the simulations); thus, the improvements in Model-IC are due to us modeling the bias in optimistic unchokes. Moreover, the free-riders' download time is quite sensitive to the seeding time, e.g., the download times of free-riders is reduced by $\approx 54\%$ when the average seeding time of contributing nodes increases from 0 to 30 min, while the download time of contributing nodes is only reduced by $\approx 18\%$. This suggests the need for re-considering appropriate use of seeding capacity, e.g., as modeled in Section IV.

In Figures 6 and 7, the model in [3] would give the same results as Model-PC, at the points where the fraction of free-riders and the seeding time are both 0. However, it would not be difficult include free riders in the model in [3], similar to the approach we have taken in Section II.

Experiment 4: Peer Set Size. Figure 8 depicts the download times of free-riders and contributing leechers as a function of a node's peer set size[8]. We consider the case with no seeding and bandwidth settings given in Table III. Observe that the peer set size has a significant effect on free-riders' download time (especially when it is small), while having little or no effect on contributing leechers. The free-riders download faster when the peer set size is smaller, e.g., the free-riders' download time is reduced by $\approx 40\%$ when the peer set size is reduced from 80 to 20. This occurs because under smaller peer set sizes, biased optimistic unchoking favors free-riders more. Model-IC captures this behavior well through biased optimistic unchoking. Model-PC does not consider peer set sizes, and thus

[8]Here we vary the peer set size of all nodes while only the free-riders' peer set size is varied in the large view exploit experiment below.

in Figure 8 it results in an upper bound for Model-IC as the peer set size approaches infinity.

IV. Applications

In this section we demonstrate the extensibility of our model by explicitly modeling a well known exploit of BT and a recently proposed fix for it[9].

Modeling an Exploit: We first show how to incorporate the so-called *large view exploit* [12], [13] into our model. The basic idea behind the large view exploit is for free-riders to increase their peer set size in order to increase the probability of being optimistically unchoked by a leecher or picked by a seed's unchoking mechanism. Theoretically, a free-rider can increase the download rate linearly with the increase in the peer set size. We can adapt both models to include large view exploit behavior of free-riders; due to lack of space, we illustrate the adaptation of Model-PC only. We do this by adjusting Equation (1) as follows. Again, let class h be the free-riding class with download rate d^h. And, let N_l^{LV} be the number of leechers in steady state under the large view exploit scheme. Specifically, we set $N_l^{LV} = \sum_{j=1}^{h-1} N_l^i + \alpha N_l^h$, where α is a function of the free-riders' desired peer set size. In the remainder of this paper, we set α to be the ratio of free-riders' desired peer set size to the peer set size of contributing leechers. Then, for contributing leecher classes, $1 \leq i \leq h-1$, we have:

$$d^i = u^i \frac{x_r}{x} + \frac{\sum_{j=1}^{h} N_l^j u^j \frac{x_o}{x}}{N_l^{LV}} + \frac{\sum_{j=1}^{h} N_s^j u^j}{N_l^{LV}},$$

$$\text{while} \quad d^h = \alpha \frac{\sum_{j=1}^{h} N_l^j u^j \frac{x_o}{x}}{N_l^{LV}} + \alpha \frac{\sum_{j=1}^{h} N_s^j u^j}{N_l^{LV}}.$$

We validate the resulting model in Figure 9 where we depict the average download rate of contributing leechers and free-riders, as a function of increasing free-riders' peer set size (due to the large view exploit). We consider the bandwidth setting in Table III, with contributing nodes having an average seeding time of 60 min, and where contributing leechers and seeds strive for a peer set of size 80 in all cases. We increase the average arrival rate to 2/min, thus sufficiently increasing the average number of nodes in the system for the large view exploit to work. Observe that free-riders can improve their performance (essentially linearly) through a larger peer set size, i.e., the large view exploit works. When the free-riders' peer set size is sufficiently large (e.g., larger than ≈ 170 in this case), they can do better than contributing leechers. Overall, our model make accurate predictions in the case of large view exploits.

Modeling an Exploit Fix: Real-world measurements, such as [2], [8], [9], [10], suggest that there exists a significant number of seeds in most torrents. And, as shown in Section III: (a) performance of free-riders is quite sensitive to seeding capacity, and (b) encouraging free-riders to contribute their capacity can improve overall system performance. Thus, it is important to consider alternative approaches to distributing seeding capacity. An approach which mitigates the large view exploit through alternative schemes for distribution of seeding capacity is given in [7]. We, again, illustrate extensibility and flexibility of our model by incorporating these schemes.

We first give a brief description of the schemes proposed in [7]. These schemes are motivated by observing that contributing leechers download slower at the beginning (when they have too few chunks to effectively participate in TFT) and at the end (when they only need very few chunks). The approach then is to prioritize the use of seeding capacity to those portions of the download process where it is most needed. The goal there is to (a) help contributing leechers while (b) hurting free-riders as they depend heavily on seeding capacity (as shown in Section III). The specifics of the schemes in [7] are as follows.

Sort-based (N): where a seed sorts its requesting neighbors based on the number of chunks they have and then unchokes the N which are furthest from the middle (based on sorting order). (Below we use the BT default of $N = 6$).

Threshold-based (K, N): Threshold-based schemes are similar, except that the N chosen are from those which have a certain percentage of the total number of chunks, e.g., in the experiments below we unchoke those nodes which have either $[0..\frac{K*100}{2}]\%$ or $[(100 - \frac{K*100}{2})..100]\%$ of the chunks.

Threshold Optimization: One parameter which can have a significant effect on performance is the threshold K. Intuitively, the smaller the value of K, the more aggressive is the degradation of free-riders' performance, but the greater is the danger of degrading contributing leechers' performance as well.

To model the effect of K, we conceptually divide the file with m chunks into two parts, of m_a and m_b chunks each, where $m_a = mK$ and $m_b = m(1-K)$. The download of the part of size m_a is assisted by seeds while the part of size m_b is downloaded without assistance from seeding capacity. We define d_a^i as the download rate corresponding to the sub-file of size m_a and d_b^i as the download rate for the sub-file of size m_b. We also define the corresponding download times as T_{la}^i and T_{lb}^i, where $T_{la}^i = \frac{m_a}{d_a^i}$ and $T_{lb}^i = \frac{m_b}{d_b^i}$. The total download time, T_l^i, for the entire file is then $T_{la}^i + T_{lb}^i$ and the average download rate for a class i node is $\frac{m}{T_l^i}$. We then have:

$$d_a^i = \frac{\sum_{j=1}^{h} q_{j,i} N_l^j u^j \frac{x_r}{x}}{N_l^i} + \frac{\sum_{j=1}^{h} o_{j,i} N_l^j u^j \frac{x_o}{x}}{N_l^i} + \frac{\sum_{j=1}^{h} N_s^j u^j}{N_{la}}$$

$$\text{and } d_b^i = \frac{\sum_{j=1}^{h} q_{j,i} N_l^j u^j \frac{x_r}{x}}{N_l^i} + \frac{\sum_{j=1}^{h} o_{j,i} N_l^j u^j \frac{x_o}{x}}{N_l^i},$$

where $N_{la} = \sum_{i=1}^{h} N_{la}^i$ and $N_{la}^i = \lambda^i T_{la}^i$, i.e., here d_a^i benefits from the seeding capacity and that seeding capacity is shared by all leechers within the threshold (N_{la}) while d_b^i only benefits from capacity due to regular and optimistic unchokes.

Figures 10 and 11 depict the download times of free-riders and contributing leechers, respectively, as a function of K, with the bandwidth settings given in Table III and the average

[9]The protocol change is used as an example; and other protocol variations can be easily incorporated.

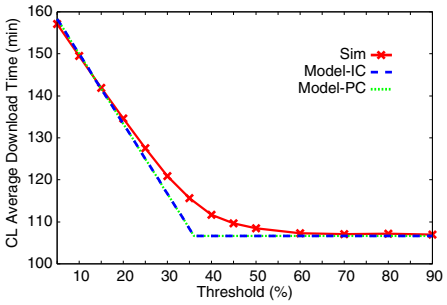

Fig. 11. Thresholds (Contributing Leechers).

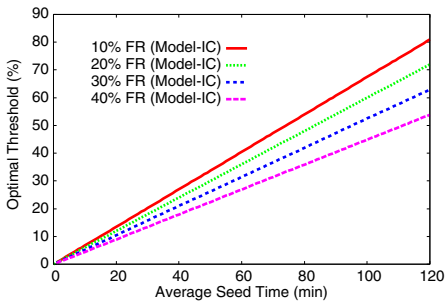

Fig. 12. Optimal Threshold

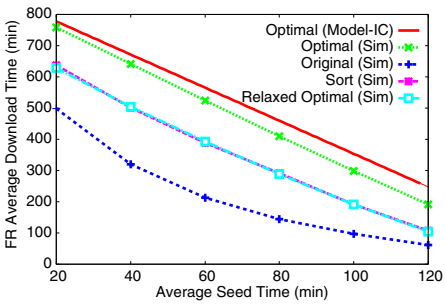

Fig. 13. Download Time using the Opt. Threshold

seed time of 60 min. Here, we observe that our models are reasonably accurate in predicting the download time, with Model-IC again being more accurate, particularly in predicting the download time of free-riders.

A number of metrics can be used to determine an optimal[10] value of K, e.g., one may want to slow down free-riders as much as possible. However, doing so may slow down the contributing leechers quite a bit, while at the same time wasting seeding capacity. Another possibility would be to optimize some notion of fairness, e.g., using fairness metrics in [3]. Another optimization objective might be to slow down free-riders as much as possible while not hurting the contributing leechers. We focus on refining this objective below.

When we examine Figures 10 and 11, we observe that there exists a knee in the threshold curves, such that when the threshold is larger than this value, reducing the threshold has no effect on contributing leechers while free-riders continue to be slowed down to some degree. Setting the threshold to this knee would be one approach to slowing down the free-riders as much as possible while not hurting contributing leechers.

This knee is actually the point where $T^i_{la} \approx 0, \forall i$ in our model. Specifically, it is the case where K is small enough for the seeding capacity to overwhelm the downloading of the m_a chunks. Reducing the threshold to a smaller value would not result in additional improvements in T^i_{la} (as well as T^i_l). The downloading of the file will then be similar to not having seeding capacity and with a file of size m_b. Thus, we observe that with further decrease in K, (i.e., increase in m_b), the download time increases linearly.

We can determine the threshold value at this knee, K', from our model as follows. When $T^i_{la} \approx 0$, $d^i_a \forall i$ is mainly dominated by the seeding capacity, we then have:

$$d^i_a \approx \frac{\sum_{j=1}^{h} N^j_s u^j}{N_{la}} = \frac{\sum_{j=1}^{h} N^j_s u^j}{\sum_{j=1}^{h} N^j_{la}} = \frac{\sum_{j=1}^{h} N^j_s u^j}{\sum_{j=1}^{h} \lambda^j T^j_{la}}$$
$$= \frac{\sum_{j=1}^{h} N^j_s u^j}{\sum_{j=1}^{h} \lambda^j \frac{m_a}{d^j_a}} = \frac{\sum_{j=1}^{h} N^j_s u^j}{\sum_{j=1}^{h} \lambda^j \frac{mK'}{d^j_a}}.$$

Since, at the knee, we have the same d^i_a for all i:

$$K' \approx min\Big(\frac{\sum_{j=1}^{h} N^j_s u^j}{\sum_{j=1}^{h} \lambda^j m}, 1\Big) = min\Big(\frac{\sum_{j=1}^{h} p^j \lambda T^j_s u^j}{\sum_{j=1}^{h} p^j \lambda m}, 1\Big)$$
$$= min\Big(\frac{\sum_{j=1}^{h} p^j T^j_s u^j}{m}, 1\Big).$$

Figure 12 depicts the optimal threshold found using this method, on a system with bandwidth settings given in Table III. We illustrate cases with $10\%, 20\%, 30\%$, and 40% of free-riding nodes in the arrivals. We observe that the optimal threshold (based on our model) increases linearly with the seeding time and smaller free-rider populations (higher contributing leecher populations) require a larger threshold. This occurs because larger thresholds are required with larger seeding capacities (i.e., longer seeding times or more contributing nodes) so that the seeding capacity is not wasted - such waste results in degraded contributing nodes' performance.

TABLE IV
DOWNLOAD TIME (MIN) USING THE OPT. THRESHOLD (CL)

	Average Seed Time (min)					
	20	40	60	80	100	120
Opt (Model-IC)	147	127	107	87	67	47
Opt (Sim)	151	133	115	96	78	59

We now focus on the case of having 20% of the arrivals being free-riders. Figure 13 and Table IV give the download times of free-riders and contributing nodes, respectively - these are computed using our model as well as simulation (where we use the optimal threshold computed from the model in the simulation settings). Addition simulation results for the original BT protocol and the sort-based scheme (described above) are also included for comparison[11]. In the simulation results, our optimal threshold setting turns out to be a bit too aggressive, as it slows down the contributing nodes as well. However, this slow down of the contributing nodes is much less sensitive than that of the free-riders when aggressive thresholds are used.

Moreover, in Figures 10 and 11, the optimal threshold value there would be larger than the one obtained from the model. Thus, we relax our threshold, i.e., we set the threshold to $K' + \theta$ (instead of K'), where θ is the relaxation factor. In Figure 13 we also show the simulation results obtained using this relaxed value with $\theta = 0.15$. Interestingly, the simulation download time using the relaxed value is very close to the download time of the sort-based approach. It can be explained by the fact that the underlying idea of the sort approach is quite similar. That is, this approach attempts to slow down free-riders as much as possible while not wasting seeding

[10]Such an optimization is not considered in [7]. To illustrate the utility of our model, here we explore an optimal threshold setting using the model.

[11]We do not include these for contributing leechers as those results are nearly identical to the Model-IC results.

capacity - i.e., sorting is similar to finding the smallest value of the threshold which can still utilize the upload capacity of the seeds. We note that the insights match the observations made from simulation experiments given in [7], whereas the computation time reduces to seconds from hours. Thus, our model provides a fast, clean and flexible approach to explore the design space of BT variations.

V. Related Work

One of the earlier BT modeling efforts [8] considers BT in two phases - initial transient phase and steady state - and propose the use of a simple Markov model (to study steady state performance) which describes the system's state using the number of leechers and seeds in the system. A number of works followed that effort, including [15] (looking at stability of BT), [9] (looking at BT's lifetime), and [6] (looking at a somewhat more detailed Markov model). A number of papers have also modeled BT-like systems, e.g., [16] models coupon replication systems and considers the make-span of a batch of nodes. A nice model of upload capacity of BT-like systems (under the assumption of it being the only constraint) is given [17]; this work also focuses on optimal make-span. However, all these works model *homogeneous* BT systems.

Since real-world torrents are typically heterogeneous, here we focus on modeling *heterogeneous* BT systems. Other models of heterogeneous systems are given in [4] and [3]. Liao et al. [4] provide a detailed model of BT with 2 heterogeneous classes of nodes, but for a *flash crowd* scenario. They consider the nodes' make-span (where all nodes join the system simultaneously) and do not include seeds. In contrast, our work focuses on steady state behavior (with a simpler model), allows for node arrivals as well as seeding, and allows an arbitrary number of node classes. Fan et al. [3] use a simple heterogeneous model for evaluating the tradeoff between performance and fairness and focus on illustrating that the number of optimistic unchokes and regular unchokes correspond to important tuning parameters, for trading off fairness and performance. We use a similar model but include seeding and free-riding behavior, which have a significant effect on BT's performance. Also, our validation results (in Section III) indicate that it is important to account for BT's *imperfect clustering* characteristics as well as a *bias in the optimistic unchoking* mechanism, both of which are also not considered in the model of [3].

We now describe several other efforts which do not focus on analytical models of BT but rather provide evidence and motivation for the modeling choices made in our work. Legout et.al [2] provide measurement studies of real BT torrents and suggest that the rarest first chunk strategy results in high diversity of chunks in the system - this provides evidence for use of simpler models, like ours, i.e., that it might be reasonable not to model efficiency of chunk exchange between peers (e.g., as is done in [6], [8], [15]). In [11], Legout et al. suggest that nodes with the same upload capacity tend to cluster with nodes of the same class. However, their results indicate (based on flash crowd arrivals), that clustering is still far from perfect (even in their closed system). Existence of imperfect clustering (or as they call it, imperfect tit-for-tat matching) is also argued in [5]. Thus, these works ([5], [11]) support the need for explicit modeling of imperfect clustering, which is done in our model. Lastly, [14] studies BT through simulations under flash crowds. We adapt their simulator for validation purposes (see Section III). Their study indicates that BT can achieve high upload bandwidth utilization, which supports our model's assumption of upload capacity being the bottleneck.

VI. Conclusions

We proposed a simple yet accurate and extensible model for BitTorrent. Our model includes (measured) characteristics of the protocol that have previously been left unmodeled, and we demonstrate the importance of including the characteristics both via our model as well as simulations. Our validation study indicates that our model is quite accurate in predicting BT performance. (We are also expanding our validation efforts through PlanetLab experiments.) Our modeling approach can aid in understanding of BT, which can in turn lead to improvements in parameter setting as well as protocol design. Our model can easily be used to answer other "what if" type questions, e.g., how the number of regular vs. optimistic unchokes affects performance (as studied in [3]), as well as effects due to different bandwidth settings, seeding times, re-evaluation intervals, history window sizes, and so on (as these are parameters in our model). Furthermore, our model can easily be extended to study protocol changes. Thus, our model can be used by the community to gain insights on the working of BT and help design improvements.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," in *P2PECON*, 2003.
[2] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *IMC*, 2006.
[3] B. Fan, D.-M. Chiu, and J. C. Lui, "The delicate tradeoffs in bittorrent-like file sharing protocol design," in *ICNP*, 2006.
[4] W.-C. Liao, F. Papadopoulos, and K. Psounis, "Performance analysis of bittorrent-like systems with heterogeneous users," in *Performance*, 2007.
[5] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent?" in *NSDI*, 2007.
[6] Y. Tian, D. Wu, and K.-W. Ng, "Modeling, analysis and improvement for bittorrent-like file sharing networks," in *INFOCOM*, 2006.
[7] A. L. Chow, L. Golubchik, and V. Misra, "Improving bittorrent: A simple approach," in *IPTPS*, 2008.
[8] X. Yang and G. de Veciana, "Service capacity in peer-to-peer networks," in *INFOCOM*, 2004.
[9] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis and modeling of bittorrent-like systems," in *IMC*, 2005.
[10] J. Bieber, M. Kenney, N. Torre, and L. P. Cox, "An empirical study of seeders in bittorrent," Duke University, Tech. Rep. CS-2006-08, 2006.
[11] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in bittorrent systems," in *SIGMETRICS*, 2007.
[12] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free riding in bittorrent is cheap," in *HotNets*, 2006.
[13] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in bittorrent networks with the large view exploit," in *IPTPS*, 2007.
[14] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving bittorrent performance," in *INFOCOM*, 2006.
[15] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *SIGCOMM*, 2004.
[16] L. Massoulie and M. Vojnovic, "Coupon replication systems," in *SIGMETRICS*, 2005.
[17] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *Journal of Scheduling*, vol. 11, no. 2, 2008.