Modelling BitTorrent-like systems with many classes of users

Wei-Cherng Liao, Netflame Technology Co., Ltd.
Fragkiskos Papadopoulos, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology
Konstantinos Psounis, Department of Electrical Engineering, University of Southern California
Constantinos Psomas, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology

BitTorrent is one of the most successful peer-to-peer systems. Researchers have studied a number of aspects of the system, including its scalability, performance, efficiency and fairness. However, the complexity of the system has forced most prior analytical work to make a number of simplifying assumptions, e.g., user homogeneity, or even ignore some central aspects of the protocol altogether, e.g., the rate-based Tit-for-Tat (TFT) unchoking scheme, in order to keep the analysis tractable.

Motivated by this, in this paper we propose two analytical models that accurately predict the performance of the system while considering the central details of the BitTorrent protocol. Our first model is a steady-state one, in the sense that it is valid during periods of time where the number of users remains fixed. Freed by the complications of user time-dynamics, we account for many of the central details of the BitTorrent protocol and accurately predict a number of performance metrics. Our second model combines prior work on fluid models with our first model to capture the transient behavior as new users join or old users leave, while modelling many major aspects of BitTorrent. To our best knowledge, this is the first model that attempts to capture the transient behavior of *many* classes of heterogeneous users. Finally, we use our analytical methodology to introduce and study the performance of a flexible token-based scheme for BitTorrent, show how this scheme can be used to block freeriders and tradeoff between higher-bandwidth and lower-bandwidth users performance, and evaluate the scheme's parameters that achieve a target operational point.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Modelling, Performance

Additional Key Words and Phrases: Heterogeneous P2P networks, BitTorrent

1. INTRODUCTION

Peer-to-peer (P2P) systems provide a powerful infrastructure for large scale distributed applications, such as file sharing. This has made them very popular. Among all P2P systems, BitTorrent seems to be the most prevalent one, since more than 50% of all P2P traffic is BitTorrent traffic [Guo et al. 2007]. It has been also estimated that BitTorrent may account for as much as 43% of all Internet traffic depending on geographical location [TorrentFreak 2009].

The BitTorrent system is designed for efficient large scale content distribution. The complete BitTorrent protocol can be found in Cohen [2003] and BitTorrent [2008]. We summarize the main functionality here. BitTorrent groups users by the file that they are interested in. In each group there exists at least one user, called seed, who has the complete file of interest. The seed is in charge of disseminating the file to other users, called

Email addresses: wcliao@gmail.com (Wei-Cherng Liao), f.papadopoulos@cut.ac.cy (Fragkiskos Papadopoulos), kpsounis@usc.edu (Konstantinos Psounis), c.psomas@cut.ac.cy (Constantinos Psomas).

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

leechers, who do not have the file. When disseminating the file, BitTorrent partitions the whole file into a large number of blocks and then the seed starts uploading blocks to its neighbors. Meanwhile, users of the group exchange the blocks they have with their neighbors. As a result, the service capacity of the system is enlarged. When a user has all the blocks of the file, he/she finishes the download process and becomes a potential seed.

There are several features making BitTorrent successful. First and foremost is the idea to partition files into large number of blocks. Another more subtle feature is the rate-based Tit-for-Tat (TFT) unchoking scheme. According to this scheme, a user uploads data to those neighbors who provide him/her the highest download rates and to one more, randomly selected neighbor, via a process called optimistic unchoking. This scheme is usually successful in penalizing freeriders, which are users that do not provide uploads to others, because they get choked. Note however that if a freerider connects to a large number of users, the aggregate rate that it can enjoy via optimistic unchoking can be substantial [Liogkas et al. 2006].

BitTorrent's success has motivated researchers to study how it performs [Guo et al. 2007; Qiu and Srikant 2004; Lo Piccolo and Neglia 2004; Wang et al. 2007; Fan et al. 2006; Clevenot et al. 2005; Tian et al. 2006; Guo et al. 2005; Chow et al. 2009; Fan et al. 2009], design additional incentive schemes for it [Liogkas et al. 2006; Hales and Patarin 2005], collect traffic measurements [Parker 2004; BigChampagne 2006; Pouwelse et al. 2005; Izal et al. 2004], and investigate fairness issues associated with it [Bharambe et al. 2006; Thommes and Coates 2005; Bin et al. 2006]. However, despite the significant interest for the system, the majority of the proposed analytical models make a number of simplifying assumptions, e.g., user homogeneity (i.e., all users have the same link capacities which is in sharp contrast to the reality of dial-up, DSL/cable, and LAN connections), or even ignore some central aspects of the protocol altogether, e.g., the TFT unchoking scheme, in order to keep the analysis tractable. This is not surprising given the system's complexity. However, such assumptions, may prohibit the resulting models to accurately capture the real system behavior [Pouwelse et al. 2005; Bharambe et al. 2006].

With this in mind, in this paper we propose two analytical models that accurately predict the performance of the system while considering many of the central details of the BitTorrent protocol. Our first model is a steady state one, in the sense that it is valid during periods of time where the number of users in the system remains fixed. Freed by the complications of user time dynamics, this model accounts for many central details of the BitTorrent protocol, including TFT and optimistic unchoking, and can accurately predict a number of performance metrics including the user download rates and the file download delays. We demonstrate that this model is applicable to one of the most common scenarios in BitTorrent: the flash crowd scenario, where most of the users join the system in a short time period just after a new file has been released [Pouwelse et al. 2005; Bharambe et al. 2006]. A preliminary version of this model has appeared in Liao et al. [2007b], where we have assumed that there exist two classes of users in the system: high-bandwidth and low-bandwidth users. Here, we show how the model can be extended to more (three) classes of users. Given that the model complexity increases fast as a function of the number of classes, and that in reality users can be mainly classified into three classes (dial-up, DSL/cable, and high speed [Saroiu et al. 2002; Bharambe et al. 2006]), we believe three classes are the best tradeoff between tractability and realism. Other differences between the model in Liao et al. [2007b] and our first model in this paper include (but are not limited to) the ability to account for multiple seeds, and an arbitrary number of concurrent connections.

Our second model builds upon prior work on fluid models for BitTorrent, see, for example, Qiu and Srikant [2004], which are traditionally used to capture the system's transient behavior and time dynamics, as new users join or old users depart. Using the techniques and intuition from our first model, we propose a fluid model that achieves the same goal as prior fluid models while accurately accounting for central properties of the BitTorrent protocol, e.g., the TFT unchoking scheme. This model can be used for steady state scenarios like the first one, but it is slightly less accurate. One of its main strengths lies in that it can also be used for non flash-crowd scenarios, where new users keep joining the system for a long time after the release of a file, and, in general, in any situation where the user population changes over time. Further, in contrast to our first model, it can easily account for any number of classes of users, while keeping the analysis simple. To our best knowledge, this is the first model that attempts to capture the transient behavior of many classes of users.

To keep our analysis tractable while focussing on central details of the BitTorrent protocol, we make two assumptions: (i) that the download link capacities of users are larger than their upload link capacities, and (ii) that the effectiveness of file sharing is perfect, i.e., that a user always has file blocks that his/her neighbors are interested in. We will explain the rationale behind these two assumptions and thoroughly study the accuracy of our models when these two assumptions are not satisfied.

Finally, as an example of how to use our analytical methodology to study variations of the basic BitTorrent scheme and make design decisions, we propose a simple and flexible token-based TFT scheme for BitTorrent. In the proposed scheme, which is inspired by our prior work on incentive schemes for P2P systems [Liao et al. 2006a; 2006b], users use tokens as a means to trade file blocks. Each user maintains a token table which keeps track of the amount of tokens his/her neighbors possess. A user increases his/her neighbor's tokens for every byte he/she downloads from the neighbor. The user decreases a neighbor's tokens for every byte he/she uploads to the neighbor under study. A user would upload a block to his/her neighbor only if the neighbor has sufficient tokens to perform the download.

We show that the above scheme can be used to tradeoff between high overall system performance and fairness to higher bandwidth users. In particular, we show that under the appropriate parameter tuning, higher bandwidth users will provide more uploads than usual to lower bandwidth users, which tends to improve the overall average download delay but worsen the perceived performance by higher bandwidth users. We also use our analytical models to compute the parameter values of the token-based scheme that achieve a target operational point. Another benefit of the token-based scheme that we discuss is its ability to effectively penalize freeriders, since users that do not offer uploads do not accumulate tokens and, in the absence of optimistic unchoking, they cannot perform any downloads.

To evaluate the performance of BitTorrent-like systems, event-driven simulators are often used, e.g., BitTorrent Simulator [2005], which try to implement all details of the BitTorrent protocol. However, such event-driven simulations of BitTorrent-like systems consisting of a large number of users can be very time-consuming and computationally intensive. Our models in this paper highlight details of the BitTorrent protocol that matter the most in performance, suggesting that lightweight simulators that do not implement all details of BitTorrent could still be used for realistic performance evaluations. In addition, we also demonstrate how our models could be used to study the performance of large BitTorent-like systems whose size may render simulation-based analysis too expensive.

To summarize, the contributions of this paper are as follows: (i) We extend our steady state model presented in Liao et al. [2007b] to account for more (three) classes of users, multiple seeds, arbitrary number of concurrent connections, and other parameters of the real protocol; (ii) we introduce a new fluid-based model that combines prior work on fluid-model analysis with techniques from our first model, to accurately capture user time dynamics and efficiently account for any number of user classes; (iii) we use the analytical methodology to assess the impact of our proposed token-based TFT scheme to freeriders and to the performance of different classes of users; (iv) we compare our models to prior work and highlight in which scenarios the assumptions made by prior work lead to sizable inaccuracies; and (v) we demonstrate how to use our models in order to study the performance of large yet realistic BitTorent-like systems whose size makes simulation-based analysis too expensive.

The rest of this paper is organized as follows: In Section 2 we briefly discuss related work. In Section 3 we review the current BitTorrent implementation in more detail, and provide a detailed description of the proposed token-based scheme. In Section 4 we present our first model, and in Section 5 our second model. In both of these sections we study the original BitTorrent system. The corresponding analysis for the token-enhanced system is similar and can be found in the online appendix in Sections A and B. In Sections 6, and C of the online appendix, we present extensive simulation results in order to validate the accuracy of our models. In Section 7 we study the accuracy of our models in scenarios where the assumptions of our analysis are not satisfied. In Section 8 we further validate our modelling methodology via experiments with real BitTorrent clients, and in Section D of the online appendix we show how our models could be used, along with real traces, in order to study the performance of large BitTorrent systems. In Section 9 we compare our models with representative models from the literature, and in Section E of the online appendix we compare our two models. Conclusions and future work directions follow in Section 10.

2. RELATED WORK

B. Cohen, the author of BitTorrent, gives a thorough introduction to the BitTorrent system in Cohen [2003]. The paper describes the BitTorrent protocol, the system architecture and the incentive scheme built in the BitTorrent system.

There is a large body of work that studies the efficiency and the popularity of BitTorrent via measurements, e.g., Liogkas et al. [2006], Guo et al. [2005], Pouwelse et al. [2005]. Qiu and Srikant [2004] proposes a fluid model to describe how the population of seeds and leechers evolves. Guo et al. [2007], Wang et al. [2007] and Fan et al. [2006] extend the above model to study BitTorrent's performance under different user behaviors and different arrival processes, and Lo Piccolo and Neglia [2004] and Clevenot et al. [2005] extend this model to study BitTorrent's performance under heterogeneous environments. Further, the recent work in Chow et al. [2009], which uses insights from our earlier work [Liao et al. 2007b], presents a model for the *steady state* performance of BitTorrent-like systems in heterogeneous environments with many classes of users, and the work in Fan et al. [2009] studies different rate assignment strategies in simplified BitTorrent-like systems in steady state, showing fundamental trade-offs between performance and fairness. Other interesting analytical results that are less relevant to our work include Tian et al. [2006] which proposes a model to study the peer distribution and uses a dying process to study the file availability in BitTorrent, Gaeta et al. [2006] which uses fluid models to study the distribution of the file transfer time in generic P2P systems that have some similar characteristics to BitTorrent, and Yang and Veciana [2004] which uses a branching process to study the service capacity of such systems.

Despite the large body of work on modeling BitTorrent's performance, the majority of the studies make a number of simplifying assumptions in order to keep the analysis tractable. For example, the studies in Guo et al. [2007], Qiu and Srikant [2004], Wang et al. [2007] and Fan et al. [2006] consider homogeneous network environments only, where users have the same link capacities. This is clearly an unrealistic assumption given Internet's heterogeneity. Those studies that consider heterogeneity make other simplifying assumptions, for example Lo Piccolo and Neglia [2004] completely ignores BitTorrent's TFT scheme, and Clevenot et al. [2005] and Fan et al. [2009] model only some aspects of it. Since BitTorrent's TFT scheme is one of the main features responsible for its great success, it is essential to accurately account for it. Finally, Chow et al. [2009] studies only the steady state behavior of heterogeneous BitTorrent systems.

Our models in this paper accurately predict the performance of BitTorrent-like systems *in both steady state and dynamic scenarios*, while considering many of the central details of the BitTorrent protocol. In particular, we consider a heterogeneous BitTorrent-like system, where users are grouped into different classes according to their link capacities and may arrive at random times, and model many of the important aspects of the system, including the TFT scheme. However, to keep our analysis tractable, we also make some simplifying assumptions. First, we assume that the download link capacities of users are larger than their upload link capacities, and second, that the effectiveness of file sharing is perfect, i.e., that a user always has file blocks that his/her neighbors are interested in. We explain the rational behind these two assumptions in Section 4, and study how accurate our models are when these two assumptions are not satisfied in Section 7.

Our proposed token-based TFT scheme is inspired by our prior work on incentive schemes for Gnutella-like P2P systems [Liao et al. 2006a; 2006b]. Both the analysis and implementation of the token-based scheme, as well as what the scheme can accomplish in BitTorrent-like systems, differs from our prior work [Liao et al. 2006a; 2006b], as we explain in detail in the next Section. Our proposed token-based TFT for BitTorrent degenerates to the block-based TFT scheme proposed in Bharambe et al. [2006] when nodes gain tokens for uploading a byte at the same rate that they use tokens to download a byte. Hence, our scheme is much more general and flexible. Further, while the work in Bharambe et al. [2006] studies the performance of the block-based TFT scheme via simulations, we extend our mathematical models to predict the performance of the token-based scheme and the block-based scheme as a special case.

Finally, our token-based scheme allows us to investigate two interesting problems related to BitTorrent. The first is about the relative performance perceived by lower and higher bandwidth users. We show how our models can be used to decide on the scheme parameters that achieve a target tradeoff between the perceived performance of lower-bandwidth and higher-bandwidth users, by making higher-bandwidth users offer more/less uploads than usual to lower-bandwidth ones. The second is about freeriders. Liogkas et al. [2006], Hales and Patarin [2005] and Sirivianos et al. [2007] have shown that despite the built-in incentive scheme in BitTorrent, skillful freeriders can still benefit from the system by connecting to many users and relying on optimistic unchoking. We show how our scheme can be used to block such skillful freeriders, motivate them to offer uploads, and as a result improve the overall performance of the system. The most relevant to this work is the one in Sherman et al. [2009], which proposes a similar scheme for providing fairness in BitTorrent-like systems, but studies its performance mostly via experiments.

3. PRELIMINARIES

3.1. Original BitTorrent

We now describe in detail the main functionality of the BitTorrent system. Recall that BitTorrent groups users by the file that they are interested in. When a user is interested in joining a group, he/she first contacts the tracker, a specific host that keeps track of all the users currently participating in the group. The tracker responds to the user with a list containing the contact information of L randomly selected peers. Typical values for L are $40 - 60$ [BitTorrent 2008]. After receiving the list, the user establishes a TCP connection to each of these L peers, which we refer to as the user's *neighbors*.

As mentioned earlier, when disseminating the file, BitTorrent partitions the whole file into a number of blocks. Neighbors exchange block availability information and messages indicating interest in blocks. The BitTorrent protocol uses a Local Rarest First (LRF) algorithm to determine which blocks to download first. It has been shown that the LRF algorithm can efficiently increase the availability of rare blocks and prevent the last block problem [Cohen 2003; Bharambe et al. 2006]. In addition, it uses a rate-based TFT scheme to determine to which neighbors a user should upload blocks to. The rate-based TFT scheme proceeds as follows: time is slotted into $T = 10$ second intervals and each such time-interval is called an *unchoking period*. At the end of each unchoking period a user makes a *choking/unchoking* decision. The choking/unchoking decision proceeds as follows: First, the user computes for each of the neighbors that are interested in downloading a block from him/her, the average download rate that he/she receives during the last $2T = 20$ seconds. Then, he/she selects to provide uploads to, i.e., to *unchoke*, a number X of his/her neighbors who provided him/her the best download rates, with ties broken arbitrarily. By default, $X = 4$. If the user chooses not to provide uploads to a neighbor, we say that the neighbor is choked. Finally, the user also randomly selects another neighbor to provide uploads to. This last (random) selection process is called *optimistic unchoking*. Hence, at any time instance a user is concurrently uploading to $Z = X + 1$ neighbors. The following rules are also adopted by the scheme.

Let's call the neighbor that was selected at the last optimistic unchoking, an *optimistic unchoking neighbor*, and suppose that the last optimistic unchoking (and hence the end of the last unchoking period) took place at time t_1 seconds. Now, suppose that the end of another unchoking period occurs at some time t_2 seconds, where $t_2 = t_1 + nT$ seconds for $n \geq 1, T = 10$. Then, if at time t_2 the optimistic unchoking neighbor belongs to the set of the X neighbors who provide the user the best download rates (and hence they will be unchoked), the user performs a new optimistic unchoking. Otherwise: (i) if $t_2 < t_1 + 3T$ seconds, the user does not choke the optimistic unchoking neighbor and does not perform a new optimistic unchoking, and (ii) if $t_2 \geq t_1 + 3T$ seconds, the user chokes the optimistic unchoking neighbor and performs a new optimistic unchoking. We call this $3T = 30$ second time-interval an *optimistic unchoking period*.

This TFT scheme discourages free-riders because they will keep getting choked if they do not provide uploads to their neighbors. Further, it gives the opportunity to new users to start downloading from the system even if they do not have enough blocks to exchange, in which case the download rate they provide is low. Finally, notice that the scheme allows a user to discover good neighbors, i.e., neighbors who provide him/her with high download rates, and exchange data with them. Therefore, users who have high upload link capacities tend to exchange data with a larger number of high capacity users. And users with low upload link capacities tend to exchange data with a larger number of low capacity users. Hence, in a sense the system is designed to be fair to each class of users.

3.2. Token-enhanced BitTorrent

The process by which a new user discovers neighbors in the proposed token-based system (which we also refer to as *token-enhanced* BitTorrent) is exactly the same as in the original BitTorrent system. Further, again, the file is partitioned into blocks and neighbors exchange block availability information and messages indicating interest in blocks.

As mentioned earlier, in the token-based scheme users use tokens as a means to trade blocks. In particular, each user maintains a token table which keeps track of the amount of tokens his/her neighbors possess. When the user uploads X_{up} bytes to a neighbor, he/she decreases the neighbor's tokens by $K_{\mathrm{down}}X_{\mathrm{up}}$. On the other hand, the user increases a neighbor's tokens by $K_{\mathrm{up}}X_{\mathrm{down}}$ if he/she downloads X_{down} bytes from the neighbor under study. Parameters K_{down} and K_{up} are selected positive constants, same for all users. Notice that a user does not have access to his/her amount of tokens since this is maintained by his/her neighbors.

Under the proposed scheme each user decides to which (of the interested) neighbors he/she will upload blocks to, every 10 seconds, which equals the unchoking period in the original BitTorrent system. In particular, every 10 seconds the user first checks which of his/her neighbors have enough tokens to perform the download of a block. If there are more than Z neighbors having enough tokens, then the user randomly selects Z of them to upload to. If Z or fewer neighbors have enough tokens the user provides uploads only to them. If a neighbor runs out of tokens while downloading from the user, then the user stops uploading to the neighbor immediately after the current block transfer is complete, and randomly selects to upload to some other neighbor who has enough tokens. Finally, we initialize the token table of each user with an amount of tokens that suffices to download one block. The reason of giving initial tokens is to allow users download data when they first join the system.

Note that $K_{\rm up}$ and $K_{\rm down}$ are relative values. Therefore, the proposed scheme actually has only one design parameter. We will show that when $K_{\rm up} = K_{\rm down}$, the token-enhanced BitTorrent system makes the aggregate upload rate of a leecher to equal its aggregate download rate, and degenerates to the block-based scheme introduced in Bharambe et al. [2006]. Further, when each leecher connects to sufficient neighbors with the same capacity, the original TFT scheme yields similar performance with the token-based scheme when $K_{\rm up} = K_{\rm down}$. We will also show that as $K_{\rm up}$ increases, the overall download delay of the token-enhanced BitTorrent system can get significantly smaller than that of the original BitTorrent system, by making higher bandwidth users connect to more lower bandwidth users than usual. Finally, we will show that the token-based scheme eliminates the problem of freeriders which exploit optimistic unchoking in the original BitTorrent, since such users cannot accumulate any tokens.

The idea behind our proposed token-based scheme is the same as in our previous work on improving performance in Gnutella-like systems by preventing freeriding [Liao et al. 2006a; 2006b]. The differences here are the following. First, since BitTorrent-like and Gnutella-like P2P systems differ significantly in many aspects, our mathematical model for the token-enhanced BitTorent is completely different from the model in Liao et al. [2006a; 2006b]. Second, here, in addition to preventing freeriders, we also study how the token-based scheme can be used to improve BitTorrent's overall performance, even if no freeriders were initially present in the system. And finally, while in Gnutella-like systems a centralized mechanism is needed for storing and updating the number of available tokens for each user, in BitTorent-like systems, as explained earlier, this is performed at each user's neighbors, which makes the scheme more scalable and robust.

4. STEADY STATE ANALYSIS

In this section we propose a mathematical model to study the performance of a BitTorrent-like system when the number of leechers and seeds in the system is assumed to remain constant over a relatively long period of time. This is the case in flash crowd scenarios [Pouwelse et al. 2005; Bharambe et al. 2006]. In the next section we study user dynamics.

We first study the user download rate and then proceed with the file download delay, which is defined as the time difference between the moment that a user (leecher) joins the system and the moment that the user downloads the complete file. The model incorporates users with heterogeneous capacities and considers the central details of the BitTorrent protocol. In Liao et al. [2007b] we have presented a model assuming two classes of users. Here, we show how it can be extended to more classes. In particular, we assume that there exist three classes of users: (i) high-bandwidth (H-BW) users, who have a high upload link capacity, (ii) medium-bandwidth (M-BW) users, who have a medium upload link capacity, and (iii) low-bandwidth (L-BW) users, who have a low upload link capacity. (The fluid-based model introduced in the next section accounts for arbitrary many classes of users. Further, as it will become apparent shortly, one of the main lessons in this section is that if one wishes to provide a detailed model for BitTorrent's performance in heterogeneous environments, the analysis becomes quite involved with only three classes of users.)

We denote by N the total number of leechers in the system and by p_{H}, p_{M}, and p_{L} the percentage of H-BW, M-BW, and L-BW leechers respectively. We present here the analysis of the original BitTorrent system. The corresponding analysis for the token-enhanced system is similar and can be found in the online appendix (Sections A, B). For brevity of exposition, we will present the detailed derivation for some of the mathematical formulas of the model, and omit the details for others, whose exact derivation follows a similar procedure. Complete derivation details for all formulas can be found in Liao et al. [2007a].

4.1. Computing the User Download Rates in the Original BitTorrent System

Let $\{n^{\mathrm{d}}_{i\mathrm{H}}(t), t \geq 0\}$ be a random process denoting the number of H-BW neighbors that leecher i is downloading from, i.e., at time t leecher i is downloading from $n^{\mathrm{d}}_{i\mathrm{H}}(t)$ H-BW neighbors. Since leechers can be either H-BW, M-BW, or L-BW and the statistics governing the leechers within each group are the same, let $\{n^{\mathrm{d}}_{\mathrm{HH}}(t), t \geq 0\}$, $\{n^{\mathrm{d}}_{\mathrm{MH}}(t), t \geq 0\}$, and $\{n^{\mathrm{d}}_{\mathrm{LH}}(t), t \geq 0\}$ be the random process denoting the number of H-BW neighbors that a H-BW, M-BW, and L-BW leecher is downloading from respectively. Our analysis is valid in periods of time where this process is stationary (or, by slightly abusing terminology, in steady state), and we work with the average value of this process denoted by $n^{\mathrm{d}}_{\mathrm{HH}}$, $n^{\mathrm{d}}_{\mathrm{MH}}$, and $n^{\mathrm{d}}_{\mathrm{LH}}$. (See Figure 2 for a graphical representation of the periods of time that we assume stationarity to hold in case of a flash crowd scenario.) In general, let n^{d}_{ij} $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$ be the average number of j-BW neighbors that an i-BW leecher is downloading from, and denote by D_{ij} the corresponding average download rates. Finally, let D_{S} be the average download rate that a leecher can receive from the seed(s). Now, if R_{downH}, R_{downM}, and R_{downL} denote the aggregate download rate of a H-BW, a M-BW, and a L-BW leecher respectively and C_{downH}, C_{downM}, C_{downL} are their corresponding download link capacities, it is easy to see that:

$$R_{\mathrm{downH}} = \min\left(n^{\mathrm{d}}_{\mathrm{HH}} D_{\mathrm{HH}} + n^{\mathrm{d}}_{\mathrm{HM}} D_{\mathrm{HM}} + n^{\mathrm{d}}_{\mathrm{HL}} D_{\mathrm{HL}} + D_{\mathrm{S}}, C_{\mathrm{downH}}\right), \tag{1}$$

$$R_{\mathrm{downM}} = \min\left(n^{\mathrm{d}}_{\mathrm{MH}} D_{\mathrm{MH}} + n^{\mathrm{d}}_{\mathrm{MM}} D_{\mathrm{MM}} + n^{\mathrm{d}}_{\mathrm{ML}} D_{\mathrm{ML}} + D_{\mathrm{S}}, C_{\mathrm{downM}}\right), \tag{2}$$

$$R_{\mathrm{downL}} = \min\left(n^{\mathrm{d}}_{\mathrm{LH}} D_{\mathrm{LH}} + n^{\mathrm{d}}_{\mathrm{LM}} D_{\mathrm{LM}} + n^{\mathrm{d}}_{\mathrm{LL}} D_{\mathrm{LL}} + D_{\mathrm{S}}, C_{\mathrm{downL}}\right). \tag{3}$$

Because all leechers in the system are equally likely to be downloading file blocks from the seeds, we can write $D_{\mathrm{S}} = C_{\mathrm{upS}}/N$, where C_{upS} is the aggregate upload link capacity of the seeds.

Our end-goal is to compute the user download delays which requires the computation of the download rates R_{downH}, R_{downM}, and R_{downL}. Hence, we need to calculate the values of the parameters n^{d}_{ij} and D_{ij} for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$. Since the BitTorrent protocol specifies the rules according to which a user chooses a neighbor to provide *uploads to*, it is more natural to work with quantities that describe the upload rather than the download dynamics, and then go from the former to the later. With this in mind, denote by n^{u}_{ij} $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$ the average number of j-BW neighbors that an i-BW leecher is uploading to, and let U_{ij} be the corresponding average upload rates. The connection between download and upload parameters is the following. First, notice that $D_{ij} = U_{ji}$ for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$. Further, in any system the total number of upload connections equals the total number of download connections. For example, the total number of upload connections provided by H-BW leechers to L-BW leechers equals the total number of download connections that L-BW leechers receive from H-BW leechers. Thus, we can write $n^{\mathrm{d}}_{\mathrm{LH}} = n^{\mathrm{u}}_{\mathrm{HL}} p_{\mathrm{H}}/p_{\mathrm{L}}$. More generally, $n^{\mathrm{d}}_{ij} = n^{\mathrm{u}}_{ji} p_j/p_i$ for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$. Therefore, what remains is to compute n^{u}_{ij} and U_{ij} for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$.

First, let R_{upH}, R_{upM}, and R_{upL} be the aggregate upload rate of a H-BW, a M-BW, and a L-BW leecher respectively, and C_{upH}, C_{upM}, and C_{upL} be the upload link capacity of

H-BW, M-BW and L-BW leechers respectively. For ease of analysis, we assume that the download link capacity of each user in the system is larger than the upload link capacity of every other user. This means that the upload links of users are always fully utilized. Note that in practice this is clearly the case for users belonging to the same class, consider, for example, DSL, Cable, and Ethernet technologies. However, it is not impossible to have a H-BW user with upload capacity larger than the download capacity of a L-BW user. As we said, we ignore this situation to keep the analysis tractable, and leave an extension of the analysis without this assumption as future work. However, we study the accuracy of our model when this assumption is not satisfied via experiments in Section 7, showing that the model can still give very good predictions, even if only the download capacity of users belonging to the same class is larger than their upload capacity. Therefore, we set $R_{\mathrm{upH}} = C_{\mathrm{upH}}$, $R_{\mathrm{upM}} = C_{\mathrm{upM}}$, $R_{\mathrm{upL}} = C_{\mathrm{upL}}$, and it is easy to see that:

$$C_{\mathrm{upH}} = n^{\mathrm{u}}_{\mathrm{HH}} U_{\mathrm{HH}} + n^{\mathrm{u}}_{\mathrm{HM}} U_{\mathrm{HM}} + n^{\mathrm{u}}_{\mathrm{HL}} U_{\mathrm{HL}}, \tag{4}$$
$$C_{\mathrm{upM}} = n^{\mathrm{u}}_{\mathrm{MH}} U_{\mathrm{MH}} + n^{\mathrm{u}}_{\mathrm{MM}} U_{\mathrm{MM}} + n^{\mathrm{u}}_{\mathrm{ML}} U_{\mathrm{ML}}, \tag{5}$$
$$C_{\mathrm{upL}} = n^{\mathrm{u}}_{\mathrm{LH}} U_{\mathrm{LH}} + n^{\mathrm{u}}_{\mathrm{LM}} U_{\mathrm{LM}} + n^{\mathrm{u}}_{\mathrm{LL}} U_{\mathrm{LL}}. \tag{6}$$

We need 15 more equations (in addition to Equations (4)...(6)) in order to compute our unknowns. (In general, if there are n classes of users, one would need to solve a system of $(n+n) \cdot n = 2n^2$ equations. This is because each class $C \in \{1...n\}$ is characterized by n variables dictating the number of users from each class that a member of class C is uploading to on average, and n corresponding upload rates.) The first three are trivial. If Z is the number of neighbors that a user in BitTorrent is uploading to at any time instance (by default $Z = 5$) then:

$$n^{\mathrm{u}}_{\mathrm{HH}} + n^{\mathrm{u}}_{\mathrm{HM}} + n^{\mathrm{u}}_{\mathrm{HL}} = Z, \tag{7}$$
$$n^{\mathrm{u}}_{\mathrm{MH}} + n^{\mathrm{u}}_{\mathrm{MM}} + n^{\mathrm{u}}_{\mathrm{ML}} = Z, \tag{8}$$
$$n^{\mathrm{u}}_{\mathrm{LH}} + n^{\mathrm{u}}_{\mathrm{LM}} + n^{\mathrm{u}}_{\mathrm{LL}} = Z. \tag{9}$$

Taking the specifics of BitTorrent into consideration, Lemma 4.1 later in this section shows how to compute $n^{\mathrm{u}}_{\mathrm{HM}}$, $n^{\mathrm{u}}_{\mathrm{HL}}$, and $n^{\mathrm{u}}_{\mathrm{ML}}$ and Lemma 4.2 shows the way to compute $n^{\mathrm{u}}_{\mathrm{MH}}$, $n^{\mathrm{u}}_{\mathrm{LH}}$, and $n^{\mathrm{u}}_{\mathrm{LM}}$. With these quantities known, one can also compute $n^{\mathrm{u}}_{\mathrm{HH}}$, $n^{\mathrm{u}}_{\mathrm{MM}}$, and $n^{\mathrm{u}}_{\mathrm{LL}}$ from Equations (7)...(9).

Before proceeding in computing the aforementioned variables, let's first write the relationships between U_{ij}'s for $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$. Since peer to-peer traffic is transferred via TCP connections, we can assume that the upload capacity of a user will be fairly shared among concurrent upload connections. (This is a working assumption, also made in many other studies of P2P systems, e.g., see Piatek et al. [2007].) Therefore, it is easy to see that:

$$U_{\mathrm{LH}} = U_{\mathrm{LM}}, \tag{10}$$
$$U_{\mathrm{LH}} = U_{\mathrm{LL}}, \tag{11}$$
$$U_{\mathrm{MH}} = U_{\mathrm{MM}}, \tag{12}$$
$$U_{\mathrm{MH}} = U_{\mathrm{ML}}, \tag{13}$$
$$U_{\mathrm{HH}} = U_{\mathrm{HM}}, \tag{14}$$
$$U_{\mathrm{HH}} = U_{\mathrm{HL}}. \tag{15}$$

4.1.1. Computing $n^{\mathrm{u}}_{\mathrm{HL}}$, $n^{\mathrm{u}}_{\mathrm{ML}}$, and $n^{\mathrm{u}}_{\mathrm{HM}}$. We start with $n^{\mathrm{u}}_{\mathrm{HL}}$, which is the average number of L-BW leechers that a H-BW leecher provides uploads to. Let L be the total number of a leecher's neighbors and assume that all of these neighbors are interested in a block that the leecher under study possesses. (It has been demonstrated that file sharing in BitTorrent is very effective, i.e., there is a high likelihood that a node holds a block that is useful to its peers, e.g., see Bharambe et al. [2006]. This is partially due to the local rarest first (LRF) block selection algorithm that BitTorrent uses to disseminate blocks. However, we also study scenarios where this is not the case in Section 7.) Further, denote by $\text{Trinomial}(N, p_1, p_2, k_1, k_2)$ the Trinomial distribution function with parameters N, p_1, and p_2, that is, $\text{Trinomial}(N, p_1, p_2, k_1, k_2) \equiv \begin{pmatrix} N \\ k_1 \end{pmatrix} \begin{pmatrix} N - k_1 \\ k_2 \end{pmatrix} p_1^{k_1} p_2^{k_2} (1 - p_1 - p_2)^{(N-k_1-k_2)}$. Then, n^{u}_{HL} is given by the following lemma:

LEMMA 4.1.

$$n^{\text{u}}_{\text{HL}} = \sum_{k=0}^{L} \sum_{j=0}^{L-k} n_1(j,k) P_1(j,k), \tag{16}$$

where:

$$n_1(j,k) = \begin{cases} \frac{L-j-k}{L-Z+1} & \textit{if } j+k \geq Z, \\ Z-k-j & \textit{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_1(j,k) =& P\{\textit{have } j \textit{ M-BW and } k \textit{ H-BW neighbors out of } L\} \\ =& \text{Trinomial}(L, p_{\text{M}}, p_{\text{H}}, j, k). \end{aligned}$$

PROOF. First recall that p_{M} and p_{H} are the percentage of M-BW and H-BW leechers in the system. Since the neighbors' list consists of a random selection of H-BW, M-BW, and L-BW leechers, it is easy to see that $P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of } L\} = \text{Trinomial}(L, p_{\text{M}}, p_{\text{H}}, j, k)$, where $j + k \leq L$. The variable $n_1(j,k)$ represents the average number of L-BW leechers that a H-BW leecher, say leecher χ, is uploading to, given that it currently has k and j H-BW and M-BW neighbors respectively. More precisely, since χ provides uploads to Z of his/her neighbors, we distinguish two cases: (i) $j + k \geq Z$, and (ii) $j + k < Z$. First, consider case (i) and recall how BitTorrent's TFT scheme works (see Section 3). It is easy to see that in this case χ may be uploading to at most one L-BW leecher at any time instance. This L-BW leecher is randomly selected (via optimistic unchoking) with probability $(L - k - j)/(L - Z + 1)$. Now consider case (ii). In this case χ is uploading to exactly $Z - k - j$ L-BW leechers at any time instance, as he/she does not have any other higher bandwidth neighbor that he/she could provide uploads to. It is now easy to see that n^{u}_{HL} is given by Equation (16). □

In a similar manner, n^{u}_{ML} can be computed as follows:

$$n^{\text{u}}_{\text{ML}} = \sum_{i=0}^{L} \sum_{j=0}^{L-i} n_2(i,j) P_2(i,j), \tag{17}$$

where:

$$n_2(i,j) = \begin{cases} \frac{i}{L-Z+1} & \text{if } j \geq Z, \\ \frac{(Z-j)i}{L-j} & \text{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_2(i,j) =& P\{\text{have } i \text{ L-BW and } j \text{ M-BW neighbors out of } L\} \\ =& \text{Trinomial}(L, p_{\text{L}}, p_{\text{M}}, i, j). \end{aligned}$$

And finally, by noticing that H-BW leechers will prefer M-BW to L-BW leechers when they do not have sufficient H-BW neighbors to upload to, it is easy to see that $n^{\mathrm{u}}_{\mathrm{HM}}$ is given by the following equation:

$$n^{\mathrm{u}}_{\mathrm{HM}} = \sum_{k=0}^{L} \sum_{j=0}^{L-k} n_3(j,k) P_1(j,k), \tag{18}$$

where:

$$n_3(j,k) = \begin{cases} \frac{j}{L-Z+1} & \text{if } k \geq Z, \\ Z-k-1 & \text{if } k < Z \text{ and } j \geq Z-k-1, \\ j & \text{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_1(j,k) =& P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of } L\} \\ =& \text{Trinomial}(L, p_{\mathrm{M}}, p_{\mathrm{H}}, j, k). \end{aligned}$$

4.1.2. Computing $n^{\mathrm{u}}_{\mathrm{LH}}$, $n^{\mathrm{u}}_{\mathrm{LM}}$, and $n^{\mathrm{u}}_{\mathrm{MH}}$. We start with $n^{\mathrm{u}}_{\mathrm{LH}}$, which is the average number of H-BW leechers that a L-BW leecher provides uploads to. For this we need to consider the details of BitTorrent's TFT mechanism. First, recall from Section 3 that the optimistic unchoking period is 30 seconds, the rate observation window is 20 seconds, and users make their choking decision every $T = 10$ seconds. Suppose that a H-BW leecher j selects a L-BW leecher i via optimistic unchoking at time t_0, as shown in Figure 1. According to BitTorrent's TFT scheme, at time $t_0 + 30$ leecher j will choke i, because i did not provide him/her with a high download rate.

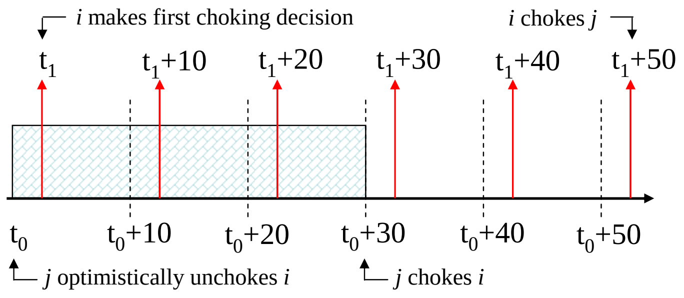


Fig. 1. Time line of optimistic unchoking and choking decision making.

Now, let's study the outcome of the choking decisions of L-BW leecher i. Suppose that this leecher makes his/her first choking decision at time t_1. Clearly, user i will not choke leecher j at t_1, $t_1 + 10$, and $t_1 + 20$ because j provides him/her with a higher download rate compared to U_{LL} (the rate by which i is downloading from a L-BW neighbor). (Notice that the probability that two or more higher bandwidth users provide uploads to the same L-BW leecher at the same time-instance is small, as the L-BW leecher is only selected by higher bandwidth users via optimistic unchoking. Therefore, i will be always downloading from at least one L-BW neighbor.) Further, leecher i will choke j at time $t_1 + 50$ because the rate observation window is 20 seconds and leecher j did not provide anything to i during the period $(t_1 + 30, t_1 + 50]$. How about $t_1 + 30$ and $t_1 + 40$? At $t_1 + 30$, the average download rate that i observes from j is $U_{\mathrm{HL}}(20 + t_0 - t_1)/20$. If this rate is larger than U_{LL}, i will not choke j. Similarly, at $t_1 + 40$, the average download rate that i observes from j is

$U_{\mathrm{HL}}(10+t_0-t_1)/20$. If this rate is larger than U_{LL}, i will not choke j. Therefore, if $N^{\mathrm{LH}}_{\mathrm{unchoke}}$ denotes the number of times that i (a L-BW leecher) did not choke j (a H-BW leecher) in the time-interval $[t_1, t_1+50]$, we can write:

$$N^{\mathrm{LH}}_{\mathrm{unchoke}} = \begin{cases} 3 & \text{if } U_{\mathrm{HL}}\frac{(2T+t_0-t_1)}{2T} < U_{\mathrm{LL}}, \\ 5 & \text{if } U_{\mathrm{HL}}\frac{(T+t_0-t_1)}{2T} \geq U_{\mathrm{LL}}, \\ 4 & \text{otherwise,} \end{cases}$$

where $T = 10$ is the duration of the unchoking period. Because users are not synchronized (and the choking decisions are taking place every $T = 10$ seconds) it makes sense to assume that t_1 is uniformly distributed between t_0 and $t_0 + T$. Hence, we can compute the average number of times $\overline{N}^{\mathrm{LH}}_{\mathrm{unchoke}}$ that i (a L-BW leecher) did not choke j (a H-BW leecher). This corresponds to a duration of $T\overline{N}^{\mathrm{LH}}_{\mathrm{unchoke}}$ seconds.

Recall that a H-BW leecher is uploading to $n^{\mathrm{u}}_{\mathrm{HL}}$ L-BW leechers on average. Therefore, considering the above scenario only, it is easy to see that at any time instance a H-BW leecher on average downloads from $n^{\mathrm{u}}_{\mathrm{HL}}T\overline{N}^{\mathrm{LH}}_{\mathrm{unchoke}}/(3T)$ L-BW leechers. And hence, the average number of H-BW leechers that a L-BW leecher provides uploads to (due to the above scenario only) is $(p_{\mathrm{H}}/p_{\mathrm{L}})n^{\mathrm{u}}_{\mathrm{HL}}T\overline{N}^{\mathrm{LH}}_{\mathrm{unchoke}}/(3T)$. We refer to this scenario, as the *optimistic unchoking reward scenario*. $n^{\mathrm{u}}_{\mathrm{LH}}$ is now given by the following lemma:

LEMMA 4.2.

$$n^{\mathrm{u}}_{\mathrm{LH}} = \sum_{i=0}^{L}\sum_{k=0}^{L-i} n_4(i,k)P_3(i,k) + \left(\frac{p_{\mathrm{H}}}{p_{\mathrm{L}}}\right) n^{\mathrm{u}}_{\mathrm{HL}} \frac{\overline{N}^{\mathrm{LH}}_{\mathrm{unchoke}}}{3}, \tag{19}$$

where:

$$n_4(i,k) = \begin{cases} \frac{k}{L-Z+1} & \textit{if } i \geq Z, \\ \frac{(Z-i)k}{L-i} & \textit{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_3(i,k) =& P\{\textit{have } i \textit{ L-BW and } k \textit{ H-BW neighbors out of } L\} \\ =& \mathrm{Trinomial}(L, p_{\mathrm{L}}, p_{\mathrm{H}}, i, k). \end{aligned}$$

PROOF. The first term of Equation (19) is derived in a similar way as Lemma 4.1 and accounts for the average number of H-BW leechers that a L-BW leecher uploads to, without considering the optimistic unchoking reward scenario. The second term of the relation accounts for the optimistic unchoking reward scenario and has been derived in the text before the lemma statement. □

Notice that in Lemma 4.1 we have not considered the optimistic unchoking reward scenario. This is because, if a L-BW leecher selects via optimistic unchoking a H-BW leecher to provide uploads to, the H-BW leecher will choke this L-BW leecher on his/her first choking decision, because the L-BW leecher does not provide him/her with a high download rate. Therefore, H-BW leechers do not provide uploads to L-BW leechers in this case (i.e., L-BW leechers are not getting any reward for optimistically unchoking H-BW leechers.)

Following the same reasoning, we can also compute $\overline{N}^{\mathrm{LM}}_{\mathrm{unchoke}}$, the average number of times that a L-BW leecher does not unchoke a M-BW leecher who has selected the former via optimistic unchoking, and $\overline{N}^{\mathrm{MH}}_{\mathrm{unchoke}}$, the average number of times that a M-BW leecher does not unchoke a H-BW leecher. Then, in a similar manner as in Lemma 4.2, $n^{\mathrm{u}}_{\mathrm{LM}}$ and

$n^{\mathrm{u}}_{\mathrm{MH}}$ are computed as follows:

$$n^{\mathrm{u}}_{\mathrm{LM}} = \sum_{i=0}^{L} \sum_{j=0}^{L-i} n_5(i,j) P_2(i,j) + \left(\frac{p_{\mathrm{M}}}{p_{\mathrm{L}}}\right) n^{\mathrm{u}}_{\mathrm{ML}} \frac{\overline{N}^{\mathrm{LM}}_{\mathrm{unchoke}}}{3}, \tag{20}$$

where:

$$n_5(i,j) = \begin{cases} \frac{j}{L-Z+1} & \text{if } i \geq Z, \\ \frac{(Z-i)j}{L-i} & \text{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_2(i,j) = & P\{\text{have } i \text{ L-BW and } j \text{ M-BW neighbors out of } L\} \\ = & \mathrm{Trinomial}(L, p_{\mathrm{L}}, p_{\mathrm{M}}, i, j). \end{aligned}$$

$$n^{\mathrm{u}}_{\mathrm{MH}} = \sum_{j=0}^{L} \sum_{k=0}^{L-j} n_6(j,k) P_1(j,k) + \left(\frac{p_{\mathrm{H}}}{p_{\mathrm{M}}}\right) n^{\mathrm{u}}_{\mathrm{HM}} \frac{\overline{N}^{\mathrm{MH}}_{\mathrm{unchoke}}}{3}, \tag{21}$$

where:

$$n_6(j,k) = \begin{cases} \frac{j}{L-Z+1} & \text{if } k \geq Z, \\ \frac{(Z-k)j}{L-k} & \text{otherwise.} \end{cases}$$

and:

$$\begin{aligned} P_1(j,k) = & P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of } L\} \\ = & \mathrm{Trinomial}(L, p_{\mathrm{M}}, p_{\mathrm{H}}, j, k). \end{aligned}$$

From Equations (4)...(21) we can now compute n^{u}_{ij} and U_{ij} for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$. And, we can relate these parameters to n^{d}_{ij} and D_{ij} as described earlier, in order to compute the average download rates using Equations (1)...(3). The computation of the user download rates in the token-enhanced system is similar and can be found in Section A of the online appendix.

4.2. Estimating the Average Download Delay

So far, we have seen how one can compute the download rates in a heterogeneous BitTorrent-like system under steady state assumptions. Now, we show how one can compute the file download delays from the corresponding rates.

As mentioned earlier, the steady state assumption makes sense in flash crowd scenarios [Pouwelse et al. 2005; Bharambe et al. 2006]. In such scenarios leechers will join the system in a short time period. As a consequence, the total number of leechers present in the system will stabilize quickly, and will remain constant for a relatively long time-period, until leechers finish their downloads and start departing the system (or becoming seeds). Figure 2 shows how the total number of leechers in a system with H-BW, M-BW, and L-BW leechers will evolve as a function of time. During the time period $(t_0, t_1]$, leechers join the system. From t_1 to t_2, all three classes of leechers are present in the system. Since H-BW leechers have higher capacities, they depart earlier, by time t_3. During the time period $(t_3, t_4]$, only M-BW and L-BW leechers are present in the system. At time t_5, all M-BW leechers depart and afterwards only L-BW leechers are present in the system. Our model computes the download rates for each class of leechers during the time interval $(t_1, t_2]$. Further, the new download rate of L-BW and M-BW leechers during the interval

$(t_3, t_4]$ (after all H-BW leechers leave the system) can be computed using our two user-class model presented in Liao et al. [2007b]. Finally, the download rate of L-BW leechers during the interval $(t_5, t_6]$ is just equal to the sum of their upload link capacity, since this is fully utilized as explained earlier, plus the download rate they receive from seeds, or, just equal to their download link capacity if this is smaller than the previous sum. Notice that our earlier analysis does not model the transient periods $(t_0, t_1]$ $(t_2, t_3]$, $(t_4, t_5]$, $(t_6, t_7]$. To compute the delays we make the assumption that $t_0 \approx t_1$, $t_2 \approx t_3$, $t_4 \approx t_5$, and $t_6 \approx t_7$. As we shall see in Section 6, these approximations do not significantly affect the accuracy of the model.

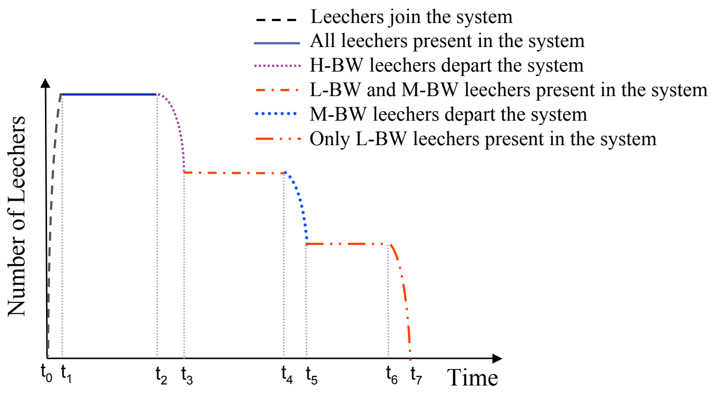


Fig. 2. Evolution of the number of leechers.

Now, let S be the file size and let T_{H}, T_{M}, and T_{L} be the average file download delay of a H-BW, a M-BW, and a L-BW leecher respectively. Clearly, $T_{\mathrm{H}} = S/R_{\mathrm{downH}}$.

Now, let's consider a M-BW leecher. Let S_{M} be the amount of data that the M-BW leecher has downloaded when all H-BW leechers were present in the system, that is, $S_{\mathrm{M}} = T_{\mathrm{H}} R_{\mathrm{downM}}$. Further, denote by R'_{downM} the new average download rate of a M-BW user after all H-BW leechers left the system, computed using our model in Liao et al. [2007b], as explained above. It is easy to see that $T_{\mathrm{M}} = T_{\mathrm{H}} + (S - S_{\mathrm{M}})/R'_{\mathrm{downM}}$.

Finally, if R'_{downL} is the new average download rate of a L-BW leecher after all H-BW leechers left the system (computed using our model in Liao et al. [2007b]), then $S_{\mathrm{L}} = R_{\mathrm{downL}} T_{\mathrm{H}} - R'_{\mathrm{downL}}(T_{\mathrm{M}} - T_{\mathrm{H}})$ is the amount of data that a L-BW leecher has downloaded until all M-BW leechers have left the system. Using the same reasoning as above, the average file download delay of a L-BW leecher is $T_{\mathrm{L}} = T_{\mathrm{M}} + (S - S_{\mathrm{L}})/(C_{\mathrm{upL}} + D'_{\mathrm{S}})$, where $D'_{\mathrm{S}} = C'_{\mathrm{upS}}/(p_{\mathrm{L}} N)$ is the average download rate from seeds. Notice that the aggregate seed capacity C'_{upS} may not be the same as before, since some H-BW and M-BW leechers might have become seeds instead of leaving the system. In other words, $C'_{\mathrm{upS}} = C_{\mathrm{upS}} + p_{\mathrm{s}}^{\mathrm{H}} p_{\mathrm{H}} N C_{\mathrm{upH}} + p_{\mathrm{s}}^{\mathrm{M}} p_{\mathrm{M}} N C_{\mathrm{upM}}$, where $p_{\mathrm{s}}^{\mathrm{H}}$ and $p_{\mathrm{s}}^{\mathrm{M}}$ denote the percentage of H-BW and L-BW leechers that become seeds respectively.

Remark 4.3. Compared to our two user-class model presented in our earlier study [Liao et al. 2007b], we can observe that the extension to three classes of users becomes significantly more complicated, if one wishes to model central properties of the BitTorrent protocol. Along the same lines, one can extend the model for more classes of users. However, given that in reality there are mainly three classes of users (dial-up, DSL/cable, and high speed [Saroiu et al. 2002; Bharambe et al. 2006]) and that the model complexity increases fast as a function of the number of classes, we believe three classes are the best tradeoff between tractability and realism, and we we do not proceed with this task here. However, we use our techniques and intuition from the above analysis to propose a second model, which ignores *some* details of the BitTorrent protocol. This enables it to efficiently account for more possible classes of users and predict *both the system steady state and transient behaviors*, with satisfactory accuracy.

5. SYSTEM TIME DYNAMICS

In the previous section we have analyzed the performance of heterogeneous BitTorrent-like systems in steady state. In particular, we have derived a mathematical model that accounts for many central details of the BitTorrent protocol and predicts the download rates, and hence the delays, of different classes of users. However, as stated, this model does not capture user dynamics and does not scale well when there exist many classes of users.

With the above in mind we propose a second, fluid-based model that can be used for scenarios where users arrive and depart at any time instance, e.g., such as in non-flash crowd scenarios. Another advantage of this model is that it scales well with increasing number of classes. The model is inspired by prior work on fluid-based analysis of BitTorrent systems [Qiu and Srikant 2004] (see Section 9 and Table I for a detailed comparison with Qiu and Srikant [2004] and other prior work) and by the techniques and rational of our first model. To ease analysis, as with the first model, we assume that the download link capacity of a user is larger than the upload link capacity of every other user in the system. We also study the accuracy of this model when this assumption is not satisfied via experiments in Section 7, showing again that the model can give good predictions, even if only the download capacity of users belonging to the same class is larger than their upload capacity. Further, to keep the analysis tractable, we assume that leechers of a particular class provide uploads to leechers of other classes only via optimistic unchoking. This implies the following: (i) we assume that a leecher of a specific class always has enough neighbors of the same class to which he/she can connect to, which is not an unrealistic assumption since the list of neighbors (L) returned by the tracker is usually large; and (ii) we do not consider the optimistic unchoking reward scenario we saw earlier. As we shall see in Section 6, these approximations do not significantly affect the model's accuracy, but they do make it slightly less accurate than our first model in steady state (see Section E in the online appendix).

As before, we present below the analysis for the original BitTorrent system. The analysis for the token-enhanced system is similar and can be found in Section B of the online appendix.

5.1. The Original BitTorrent System

We say that two users, which can be either leechers or seeds, are in the same class if they have the same link capacities, and we let $\mathbf{G} = \{1, \ldots, K\}$ be the set of user classes in the system. Denote by $x_j^{\mathrm{l}}(t)$ and $x_j^{\mathrm{s}}(t)$ respectively, the number of class-j leechers and seeds in the system at time t. Let μ_j be the service rate of a class-j user, which is defined as the rate by which the user can upload a file to other users. Given the file size S and the upload link capacity of class-j users C_{up}^j, $\mu_j = C_{\mathrm{up}}^j/S$. Further, let $\mu_{\mathrm{s}}(t)$ be the aggregate service rate provided by all seeds in the system at time t. That is, $\mu_{\mathrm{s}}(t) = \sum_{j=1}^{K} \mu_j x_j^{\mathrm{s}}(t)$. Also, let $\epsilon_{\mathrm{s}i}(t)$ and $\epsilon_{ji}(t)$ be the portion of the aggregate service rate provided by all seeds and class-j leechers respectively, to all class-i leechers at time t. Finally, denote by $R_i(t)$ the aggregate service rate that all class-i leechers receive at time t. Considering the fact that class-i leechers cannot download faster than their download link capacity, C_{down}^i, it is easy to see that:

$$R_i(t) = \min\left(\sum_{j=1}^{K} x_j^{\mathrm{l}}(t)\mu_j\epsilon_{ji}(t) + \mu_{\mathrm{s}}(t)\epsilon_{\mathrm{s}i}(t), x_i^{\mathrm{l}}(t)\frac{C_{\mathrm{down}}^i}{S}\right). \tag{22}$$

Notice that $R_i(t)$ is also the departure rate of class-i leechers, and that their average file download rate is $R_i(t) \times S$. Now, let $\lambda_i(t)$ be the arrival rate of class-i leechers, and p_{s}^i be the probability that a class-i leecher will stay in the system after he/she downloads the file (i.e., the probability of becoming a seed). Further, let γ_i be the rate at which class-i seeds leave the system. Then, the population of class-i leechers and seeds in the system is described by the following differential equations:

$$x_i^{\mathrm{l}\prime}(t) = \lambda_i(t) - R_i(t), \tag{23}$$
$$x_i^{\mathrm{s}\prime}(t) = R_i(t)p_{\mathrm{s}}^i - \gamma_i x_i^{\mathrm{s}}(t). \tag{24}$$

As before, since all leechers in the system are equally likely to be downloading from the seeds, $\epsilon_{\mathrm{s}i}(t) = [x_i^{\mathrm{l}}(t)/\sum_{n=1}^{K} x_n^{\mathrm{l}}(t)]\mu_{\mathrm{s}}(t)$, $\forall i \in \mathbf{G}$. Further, recall from the previous section that leechers are inclined to exchange file blocks with other leechers that belong in their class, due to the rate-based TFT scheme. Also, by our earlier assumption, a class-i leecher can receive uploads from a leecher of some other class-j only via optimistic unchoking. Since users randomly select a neighbor for optimistic unchoking, the probability that the selected neighbor is of class-i equals $x_i^{\mathrm{l}}(t)/\sum_{n=1}^{K} x_n^{\mathrm{l}}(t)$. Further, since a class-j leecher concurrently uploads to Z neighbors, the rate that a class-i leecher can receive from it is μ_j/Z, i.e., a portion $1/Z$ of j's upload rate goes to the class-i leecher.

We can now state the following lemma for $\epsilon_{ji}(t)$, whose proof follows immediately from the above arguments:

LEMMA 5.1.

$$\epsilon_{ji}(t) = \begin{cases} \frac{x_i^{\mathrm{l}}(t)}{\sum_{n=1}^{K} x_n^{\mathrm{l}}(t)}\frac{1}{Z} & \textit{if } i \neq j, \\ 1 - \sum_{i=1, i\neq j}^{K} \epsilon_{ji}(t) & \textit{otherwise.} \end{cases} \tag{25}$$

For a system with K classes of users we have $2K$ variables ($\{x_i^{\mathrm{l}}(t)\}$, $\{x_i^{\mathrm{s}}(t)\}, i \in \mathbf{G}$) and $2K$ differential equations (two for each class, like Equations (23) and (24)), that dictate the evolution of these $2K$ variables. Therefore, we can solve using mathematical tools (e.g., such as Mathematica [Wolfram Mathematica]) this system of equations to study how the user population ($\{x_i^{\mathrm{l}}(t)\}$, $\{x_i^{\mathrm{s}}(t)\}, i \in \mathbf{G}$) and the leecher departure rates ($\{R_i(t)\}, i \in \mathbf{G}$) evolve with time. And, of course, we can compute the corresponding download delays of each class of leechers as a function of time because the user download delay is the reciprocal of the user departure rate.

Remark 5.2. Compared to our first model, our second model is much simpler, and at the same time, more general. The key approximation that enables one to significantly simplify the analysis of BitTorrent-like systems is to ignore the optimistic unchoking reward scenario. As we will demonstrate next, this does not lead to large inaccuracies.

6. EXPERIMENTS

We use an event driven BitTorrent simulator by Microsoft Research [BitTorrent Simulator 2005] to validate our analysis. The detailed simulator description can be found in Bharambe et al. [2006]. In addition, we implement in the simulator the proposed token-based scheme in order to study its impact on the system performance. The simulation results for the token-based scheme are found in the online appendix (Sections C.1, C.2) where we also show how the scheme can be used to tradeoff between high overall system performance and fairness to higher bandwidth users, and prevent free-riding (Section C.3). Below, we present the simulation results for the original BitTorrent system.

6.1. Steady State Performance Prediction and Flash Crowd Scenarios

To validate the model of Section 4, we simulate a flash crowd scenario where $N = 200$ leechers join the system within 20 seconds. Leechers leave the system as soon as they finish their downloads. We simulate the system until all leechers depart. Since here we are interested in the steady state, to avoid the rampup period at the beginning of the simulation we randomly assign each user 1% of the blocks of the file. Other simulation settings are: (i) there is only one seed in the system and the upload link capacity of the seed is 800Kbps, (ii) the file size is 300MB and the block size is 512KB, (iii) the maximum number of concurrent upload transfers is 5 (the default value), (iv) the percentage of L-BW, M-BW, and H-BW leechers are $p_{\mathrm{L}} = 0.5$, $p_{\mathrm{M}} = 0.35$, and $p_{\mathrm{H}} = 0.15$ respectively, and (v) $C_{\mathrm{upH}} = 600$Kbps, $C_{\mathrm{upM}} = 250$Kbps, and $C_{\mathrm{upL}} = 100$Kbps. The download link capacity of all leechers is set to 600Kbps.

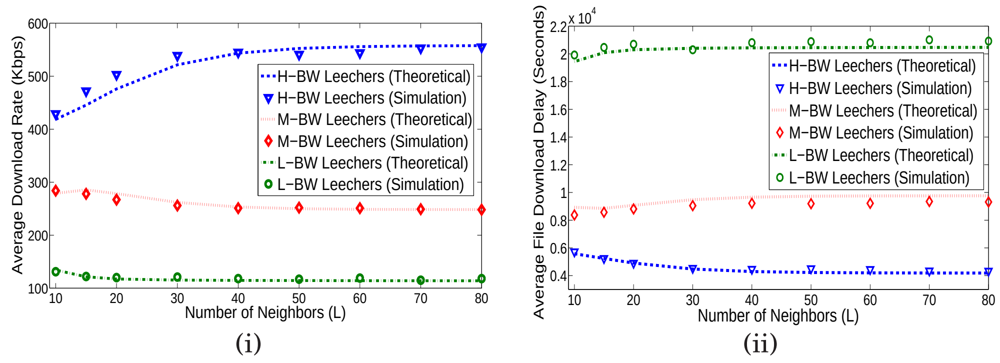


Fig. 3. (i) Average download rate for H-BW, M-BW, and L-BW leechers, (ii) Average file download delay for H-BW, M-BW, and L-BW leechers.

Figure 3(i) shows the download rates for H-BW, M-BW, and L-BW leechers with respect to the number of neighbors (L). These results correspond to the period $(t_1, t_2]$ in Figure 2, where all three classes of leechers are present in the system. First, we observe from Figure 3 that our mathematical model is quite accurate. Second, notice that the download rate of H-BW leechers increases and the download rate of M-BW and L-BW leechers decreases as L increases. This is because when L is small H-BW leechers cannot find enough H-BW peers to upload to, and thus they have to provide uploads to more M-BW and/or L-BW leechers. As L increases there are more H-BW leechers to upload to, and thus there is no need to upload to L-BW and/or M-BW leechers. Finally, Figure 3(ii) shows the average file download delay. We observe that our model is quite accurate in predicting the average file download delay for all the three classes of leechers.

6.2. Predicting System Time Dynamics and Non-flash Crowd Scenarios

We now simulate a more dynamic BitTorrent system where new leechers keep joining the system at random times according to a Poisson process, and after finishing their downloads, either depart the system or remain there for a while as seeds. This is in order to demonstrate how accurate the model of Section 5 is.

We again consider three classes of users: H-BW, M-BW, and L-BW users. The arrival rates of the three classes of leechers are different and change during the simulation as

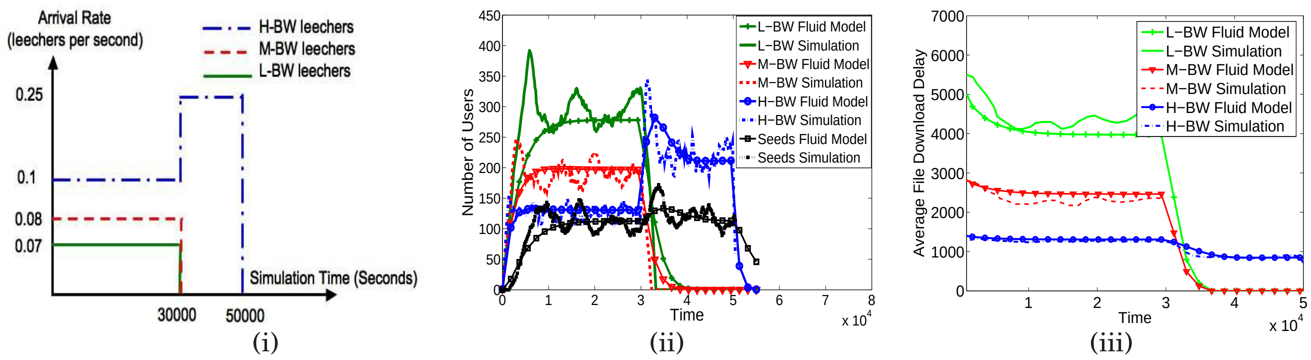


Fig. 4. (i) Leecher arrival rate, (ii) Number of users in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers.

shown in Figure 4(i). In Section 7 we also study scenarios where the arrival rates of leechers are decreasing with time. The file size is 100MB and $L = 60$. Finally, 15% of leechers stay in the system for 3000 seconds after they download the file. That is, $p_{\mathrm{s}}^{\mathrm{L}} = p_{\mathrm{s}}^{\mathrm{M}} = p_{\mathrm{s}}^{\mathrm{H}} = 0.15$ and $\gamma_{\mathrm{H}} = \gamma_{\mathrm{M}} = \gamma_{\mathrm{L}} = 1/3000$ (recall from Section 5 the definition of these parameters). All other simulation parameters are the same as before.

Figure 4(ii) shows how the number of users in the system evolves over time. From the plot we can observe that the proposed fluid model can capture the behavior of the system pretty well. Further, note that at the beginning of the simulation the number of leechers in the system is small and hence the system's service capacity, which is the aggregate service rate of all users in the system, is also small. Therefore, initially the leecher departure rate is smaller than the leecher arrival rate and this is why the number of leechers in the system initially increases. However, as the number of leechers in the system increases, the system's service capacity and hence the leecher departure rate also increase. After some time elapses, the leecher departure rate catches up with the leecher arrival rate and the system reaches its steady state, where the number of leechers in the system stabilizes. The evolution of the total number of seeds follows the evolution of leechers as expected, since some leechers, after finishing their downloads, remain to the system as seeds for some time before departing. Notice that we have intentionally stopped the arrivals of L-BW and M-BW leechers and increased the arrival rate of H-BW leechers at time 30000 in Figure 4(i). From Figure 4(ii) we can see that our fluid model can also capture this transition quite accurately.

The discrepancies between theoretical and simulation results at the beginning of the simulation in Figure 4(ii) are because the model does not consider the fact that leechers initially require a large amount of time to finish their downloads, and hence to depart the system. In particular, Equation (23) does not consider the fact that initially the leecher departure rate may be zero, but instead, it always assumes that this is strictly positive. Since the leecher departure rate is initially overestimated in the fluid model, the rate by which the number of leechers in the system increases is lower than the one in the simulation.

Finally, Figure 4(iii) shows simulation and theoretical results for the average file download delay for H-BW, M-BW, and L-BW leechers. We can observe again that our fluid model is, in general, quite accurate. The discrepancies between theoretical and simulation results in the plot are primarily due to ignoring the optimistic unchoking reward scenario. As a result: (i) the download delays of higher bandwidth leechers may be overestimated, since we ignore the download rate rewards that these leechers receive when they optimistically unchoke lower bandwidth leechers, and (ii) the download delays of lower bandwidth leechers may be underestimated, since lower bandwidth leechers no longer spend a portion of their upload capacity for rewarding higher bandwidth leechers (as occurs in reality), but instead, they use this portion for uploading to other lower capacity leechers. However, as expected, these discrepancies are relatively small.

In Section E of the online appendix we compare our two models, and in Section D we also demonstrate how we could use them, along with real traces, in order to study the performance of large BitTorrent-like systems.

7. WHEN THE ASSUMPTIONS OF THE MODEL ARE NOT SATISFIED, AND OTHER LEECHER ARRIVAL RATES

BitTorrent is a complicated protocol. As mentioned, in order to ease our analysis while focusing on central properties of the protocol (such as the TFT mechanism), we have made two main assumptions: (i) that the download link capacity of each user in the system is larger than the upload link capacity of every other user; and (ii) that file sharing is very effective, i.e., that the neighbors of a user are always interested in a block that the user under study possesses. In this section we investigate whether our models can still give good predictions when these two assumptions are not satisfied.

We first focus on the first assumption. To satisfy this assumption in the experiments we performed in Sections 6.1 and 6.2 we have set the download link capacity of each user to 600Kbps. However, as explained in Section 4, in practice it is only realistic to assume that the download link capacity of users that belong to the *same* class is larger than their upload link capacity. But notice that since users of the same class tend to exchange more data with other users belonging in their own class, we expect our models to still perform well even if only this condition is satisfied. To verify this we repeat the same experiment as in Figure 3, with the difference that now the download link capacities of H-BW, M-BW, and L-BW leechers are respectively $C_{\text{downH}} = 1200$Kbps, $C_{\text{downM}} = 500$Kbps, $C_{\text{downL}} = 200$Kbps. (Recall that $C_{\text{upH}} = 600$Kbps, $C_{\text{upM}} = 250$Kbps, and $C_{\text{upL}} = 100$Kbps). The results are shown in Figure 5(i) and indicate that our first model is still very accurate, as expected. As we show shortly, our second model is also pretty accurate in this case as well.

We now move to the second assumption. As shown in Legout et al. [2007] this assumption can be violated in a flash crowd scenario with a single seed, if the seed is underprovisioned, i.e., it has a low upload link capacity. This is because a slow seed may not be fast enough in disseminating file blocks into the system, and situations may arise where a user may not have blocks that his/her neighbors are interested in. (A similar situation in a flash crowd scenario may arise if one keeps the seed upload link capacity fixed but starts increasing the number of leechers [Al-Hamra et al. 2009]). To investigate this scenario, in Figures 5(ii) and 5(iii), we repeat the same experiment as in Figure 5(i) but with lower upload link capacities for the seed, 250Kbps and 100Kbps respectively. The behavior in these figures agrees with the observations in Legout et al. [2007] and our first model in this case is not accurate, especially for the highest bandwidth users, which was also expected. However, we note that an initial underprovisioned seed may not be a problem in non-flash crowd scenarios, where most users do not join the system in a short time after the release of a file. To verify this statement, we repeat the same experiment as in Figure 4, but with an initial seed with upload link capacity 250Kbps. We also assign to leechers the same download link capacities as in Figure 5 in order to also verify that our second model performs well in this case too. The results are shown in Figure 6, where we observe that the model can still describe the behavior of the system pretty well.

To summarize, our models can still perform pretty well, even if only the download capacity of users that belong to the same class is larger than their upload capacity. However, in scenarios where users may not always have blocks that their neighbors are interested

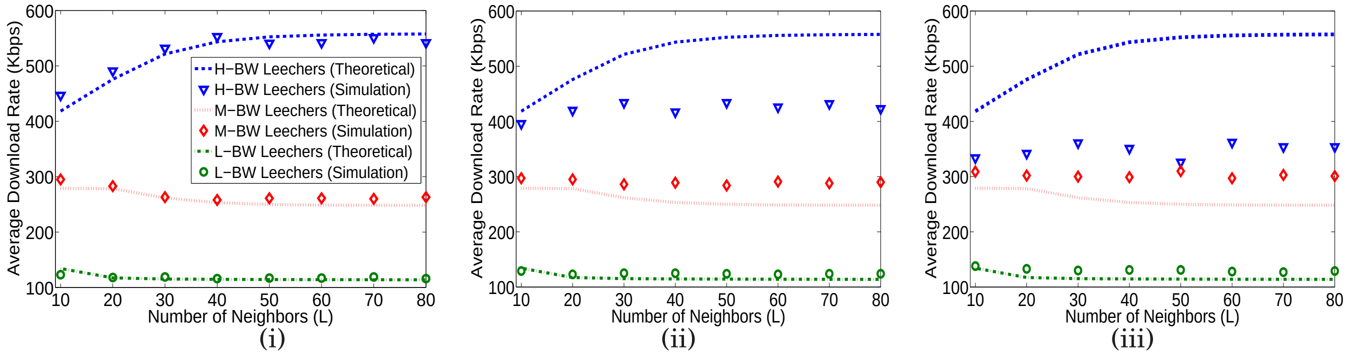


Fig. 5. Average download rate for H-BW, M-BW, and L-BW leechers with corresponding download link capacities $C_{\text{downH}} = 1200$Kbps, $C_{\text{downM}} = 500$Kbps, $C_{\text{downL}} = 200$Kbps, and seed upload link capacity: (i) 800Kbps, (ii) 250Kbps, (iii) 100Kbps. The rest of the parameters are the same as in Figure 3.

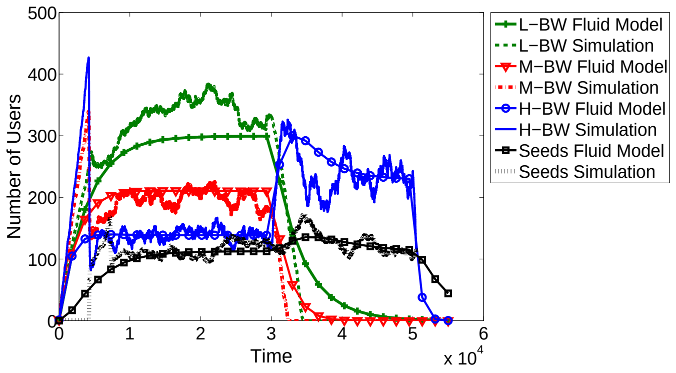


Fig. 6. Number of users in the system. The corresponding download link capacities are $C_{\text{downH}} = 1200$Kbps, $C_{\text{downM}} = 500$Kbps, $C_{\text{downL}} = 200$Kbps, and the initial seed upload link capacity is 250Kbps. The rest of the parameters are the same as in Figure 4.

in, such as in flash crowd scenarios with a relatively slow initial seed, we have seen that our first model may not be accurate, especially for higher bandwidth users (cf., Figure 5).

We finally note that our fluid model does not make any assumptions on the arrival rate of leechers. Indeed, by looking at Equation (23), this can be any function of time t. To demonstrate this we repeat the same experiment as in Figure 4, but when the arrival rate of leechers $\lambda(t)$ is an exponentially decreasing function of t, in accordance with the observations in Guo et al. [2005]. In particular, we set $\lambda(t) = \lambda_0 e^{-t/60000}$, with $\lambda_0 = 0.1, 0.08, 0.07$ for H-BW, M-BW and L-BW leechers respectively, see Figure 7(i). We also set $C_{\text{downH}} = 1200$Kbps, $C_{\text{downM}} = 500$Kbps, $C_{\text{downL}} = 200$Kbps. The rest of the parameter values are the same as in Figure 4. Figures 7(ii) and 7(iii) show the results, where we observe again that in general our fluid model performs pretty well. In Figure 7(ii) the number of leechers in the system first increases and then decreases, which is expected since the arrival rate of leechers decreases with time. We do not show the number of seeds in the system in order to avoid clutter, and since the observations are the same as before. The small underestimation in the delay of L-BW users is due to the same reasons explained in Section 6.2.

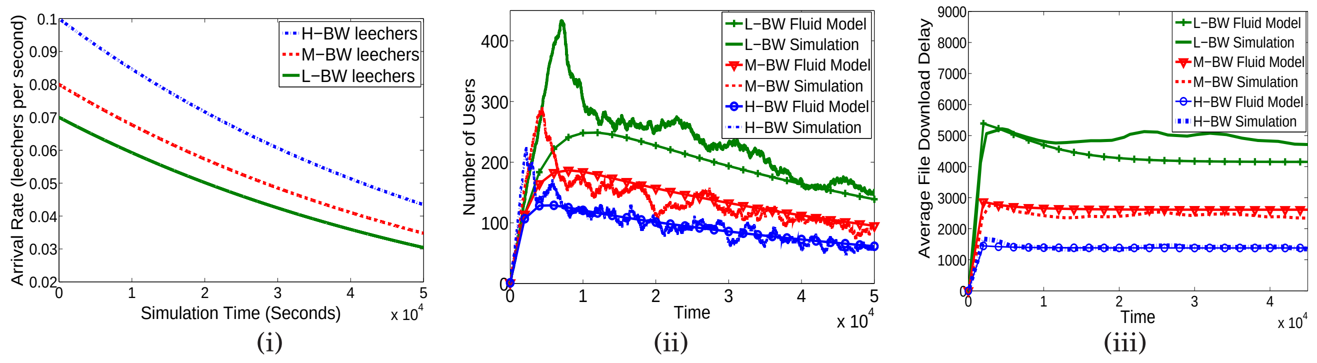


Fig. 7. (i) Decreasing leecher arrival rate, (ii) Number of leechers in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers. The download link capacities are $C_{\mathrm{downH}} = 1200\mathrm{Kbps}$, $C_{\mathrm{downM}} = 500\mathrm{Kbps}$, $C_{\mathrm{downL}} = 200\mathrm{Kbps}$. The rest of the parameters are the same as in Figure 4.

8. EXPERIMENTS WITH REAL BITTORRENT CLIENTS

To further verify the realism of our modelling methodology we also perform experiments using real BitTorrent clients. In particular, following the suggestions in Rao et al. [2010], we perform experiments on the Nephelae cloud computing cluster of the University of Cyprus [Nephelae 2012]. The machines of the cluster are running Linux (Ubuntu 11.04) as their operating system, and have an 8-core processor and 8GB of RAM. The tracker of the torrent and the seed run on different machines from the leechers. Shell scripts are used to initiate and monitor the experiments and to store useful information about each leecher, e.g., bytes downloaded, download delay, etc. The BitTorrent client used is BitTornado [BitTornado 2012], which is an improved version of the original BitTorrent client. The results are shown in Figure 8. In Figure 8(i), all the parameters are the same as in Section 6.1 apart from the number of leechers which for this experiment is $N = 100$. We observe from Figure 8(i) that the average download delays for each class of leechers are very close to our theoretical predictions, further validating our first model. In Figure 8(ii) all the parameters are the same as in Section 6.2, apart from the fact that H-BW, M-BW and L-BW leechers arrive respectively at rates $0.02, 0.016, 0.014$ leechers/second, until time 30000 seconds (and then stop). We observe again that our second model (the fluid model) can predict the number of H-BW, M-BW and L-BW leechers pretty well. The model can also predict the number of seeds in the system as before, which is not shown in the figure in order to avoid clutter.

9. COMPARISON WITH OTHER MODELS

To highlight the contributions of this paper we now compare our models with two of the most representative earlier results in the literature: Qiu and Srikant [2004] and Clevenot et al. [2005]. The reason of choosing these two results is that Qiu and Srikant [2004] presents the first mathematical model for BitTorrent systems and Clevenot et al. [2005] is one of the most representative works that considers *transient* behavior in heterogeneous environments. We refer to the model proposed in Qiu and Srikant [2004] as the QS's model and to the model proposed in Clevenot et al. [2005] as the CNR's model. The comparison is summarized in Table I.

From Table I we see that among all models, our first model is the most inclusive one in terms of modelling BitTorrent details. In particular, it models all central details of a BitTorrent-like system, except from the system time dynamics and the effectiveness of file sharing. More precisely, it considers the number of concurrent upload connections a user may have (Z), the number of neighbors to which a user might be connected (L), net-

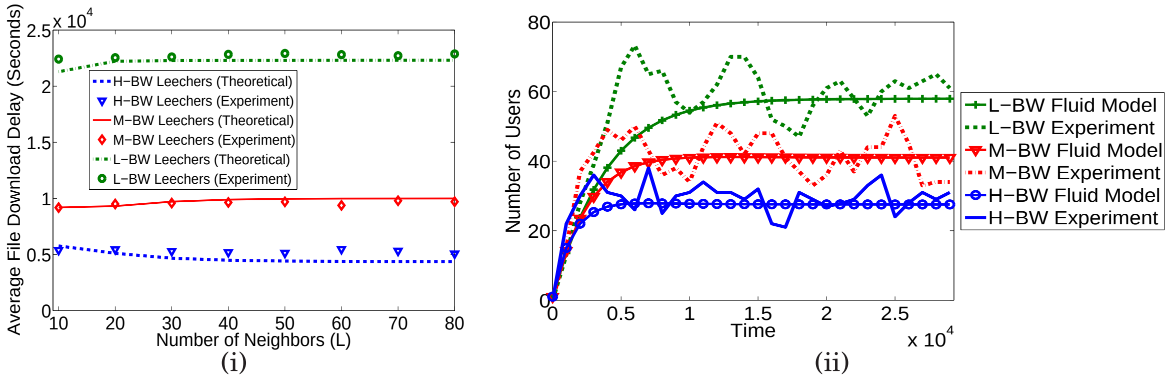


Fig. 8. (i) Average file download delay for H-BW, M-BW and L-BW leechers in experiments with real BitTorrent clients vs. theoretical predictions. The number of leechers is 100 and the rest of the setup is the same as in Section 6.1. (ii) Number of leechers in the system in experiments with real BitTorrent clients vs. theoretical predictions. The H-BW, M-BW and L-BW leechers arrive respectively at rates $0.02, 0.016, 0.014$ leechers/second, until time 30000 seconds. The rest of the setup is the same as in Section 6.2.

Table I. Model Comparison

System details captured by the Model	Model 1	Model 2	CNR's model	QS's model
number of concurrent uploads (Z)	Yes	Yes	No	No
number of neighbors (L)	Yes	No	No	No
number of user classes (heterogeneity)	3	many	2	1
effect of the TFT scheme and of optimistic unchoking	Yes	Yes	Partial	No
optimistic unchoking reward scenario	Yes	No	No	No
time dynamics	No	Yes	Yes	Yes
download link capacities larger than upload link capacities	Yes	Yes	Yes (per user class)	Yes
effectiveness of file sharing	No	No	Yes	Yes

work heterogeneity, the performance effect of BitTorrent's TFT scheme and of optimistic unchoking, as well as the optimistic unchoking reward scenario.

On the other hand, the QS's and CNR's models (as well as our second model), consider the system's time dynamics but under some approximations/simplifications. The QS's model considers system time dynamics in a homogeneous BitTorrent system, where all users have the same capacities. The CNR's model incorporates network heterogeneity (two classes of users) into the QS's model. It also attempts to model the effect of the optimistic unchoking and of the TFT scheme. However, it does not accurately capture how these affect system performance in realistic scenarios. In particular, the model assumes that users provide to other users a *static*/*fixed* proportion of their upload link capacity, in the sense that this proportion remains constant over time. But the actual computation of this proportion is much more involved, as we have shown in this paper. One must consider, for example, the exact number of concurrent upload connections (Z), the peer capacity distribution, the different user arrival rates and how these might change over time, etc. Among the three models that capture system time dynamics, our second model incorporates the most details of a BitTorrent system. In particular, it accounts for all the main protocol details, except from the variability in the number of neighbors that a user might be connected to (L), the optimistic unchoking reward scenario, and the effectiveness of file sharing. To our best knowledge, this is also the first model that considers the time dynamics of an arbitrary number of user classes in heterogeneous environments.

Both of our models and QS's model make the assumption that the download link capacity of a user is larger than (or equal) to the upload link capacity of every other user.

However, as mentioned, in a heterogeneous environment such an assumption is realistic only for users belonging to the same class. This fact is taken into consideration only in the CNR model, which only assumes that the download capacity of users belonging to the same class is larger than their upload capacity. However, as we have seen and explained, the above assumption does not significantly affect the accuracy of our models. Further, both QS and CNR consider the effectiveness of file sharing via a parameter $\eta \in [0, 1]$, which is the probability that a user has a file block that his/her neighbors are interested in. In our models, we assume that $\eta = 1$, i.e., that file sharing is very effective in BitTorrent, in accordance to the observations in Qiu and Srikant [2004]. However, in scenarios where this is not the case [Legout et al. 2007], we have seen that our first model can lead to inaccuracies.

Finally, we highlight scenarios where prior models lead to sizable inaccuracies. We consider a BitTorrent system with two classes of leechers where L-BW and H-BW leechers join with rate 0.075users/sec and leave the system as soon as they finish their downloads. Further, we have $C_{\text{upL}} = 100$Kbps, $C_{\text{downL}} = 200$Kbps, $C_{\text{upH}} = 600$Kbps, $C_{\text{downH}} = 1200$Kbps, and a seed with upload link capacity 250Kbps. All other parameters are the same as in Section 6.2.

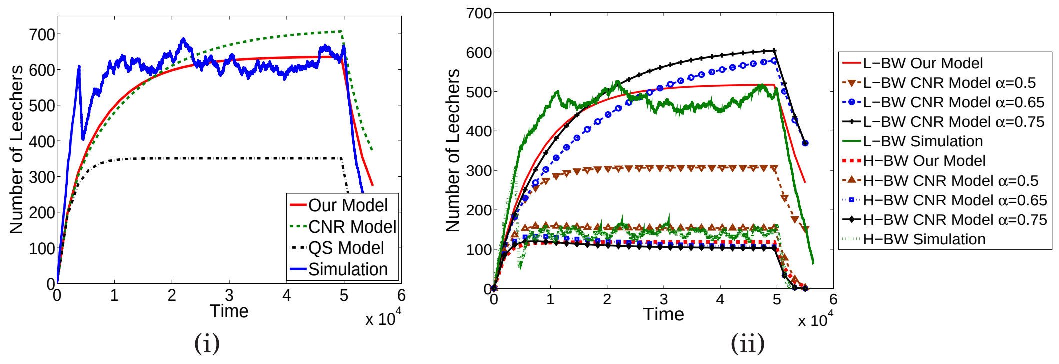


Fig. 9. (i) Inaccuracy due to the homogeneity assumption, (ii) Inaccuracy due to the static resource allocation assumption.

We compare simulation results with theoretical results obtained by our fluid model, the QS's model, and the CNR's model. Because the QS's model assumes users' homogeneity, we compute the average upload and download link capacities and use these values in our calculations whenever we use this model. Figure 9(i) shows the total number of users in the system as a function of time. We observe from the plot that, as expected, the homogeneity assumption (made by the QS's model) results in significant inaccuracies while our model and CNR's model can accurately capture simulation results. In Figure 9(ii) we present the population of H-BW and L-BW users in the system. As mentioned before, the CNR's model assumes that users provide to other users a *static/fixed* proportion of their upload link capacity, denoted in the model by parameter α. Because Clevenot et al. [2005] only provides a range of feasible α values rather than an exact one, equal to $0.5 \leq \alpha \leq 1$ in our scenario, we pick three representative values from the feasible range and plot the result. We emphasize that our model does not require to manually tune any such parameter. We observe from the plot that the static resource allocation assumption (made by the CNR's model) leads to inaccuracies for all three different values of α, especially for L-BW users, while our model can predict simulation results pretty well. For the CNR model in Figure 9(i) we used $\alpha = 0.75$.

10. CONCLUSION AND FUTURE WORK

In this paper we have proposed two mathematical models to study the performance of heterogeneous BitTorrent-like systems under different scenarios. The first model is very accurate in predicting the system performance in steady state scenarios. The second model can also be used to study time dynamics, it is simpler and more general, but it is also a bit less accurate. We also propose and analyze a flexible token-based TFT scheme that can be used to suppress the performance of freeriders, as well as tradeoff between overall system performance and fairness of high-bandwidth users, see online appendix.

As future work, we plan to incorporate the efficiency of file sharing into our models and relax the assumption that the download link capacities of users are larger than their upload link capacities. Further, we plan to use the models to address practical questions related to the design and operation of BitTorrent-like systems, e.g., to find the protocol parameter values (number of neighbors (L), number of uploads connections (Z), etc.) that yield the best download rates and delays, given, for example, the peer distribution, and the capacities and arrival rates of users. Finally, it would be very interesting to incorporate into our models notions of network proximity between peers in order to be able to analyze enhanced, topology-aware BitTorrent protocols, like the one introduced in Ren et al. [2010], and to investigate whether similar observations like the ones in this paper hold there as well.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

AL-HAMRA, A., LIOGKAS, N., LEGOUT, A., AND BARAKAT, C. 2009. Swarming overlay construction strategies. In *Proceedings of International Conference on Computer Communications and Networks, San Francisco, CA, USA*.

BHARAMBE, A., HERLEY, C., AND PADMANABHAN, V. 2006. Analyzing and improving bittorrent performance. In *Proceedings of IEEE INFOCOM, Barcelona, Spain*.

BIGCHAMPAGNE. 2006. BigChampagne Media Measurement. `http://www.bigchampagne.com/`.

BIN, F., CHIU, D.-M., AND LUI, J. C. 2006. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Proceedings of International Conference on Network Protocols (ICNP), Santa Barbara, CA, USA*.

BITTORNADO. 2012. `http://www.bittornado.com/`.

BITTORRENT. 2008. BitTorrent Protocol Specification. `http://www.bittorrent.org/beps/bep_0003.html`.

BITTORRENT SIMULATOR. 2005. Microsoft Research Simulator for the BitTorrent Protocol. `http://research.microsoft.com/en-us/downloads/20d68689-9a8d-44c0-80cd-66dfa4b0504b/`.

CHOW, A. L., GOLUBCHIK, L., AND MISRA, V. 2009. Bittorrent: An extensible heterogeneous model. In *Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil*.

CLEVENOT, F., NAIN, P., AND ROSS, K. 2005. Multiclass p2p networks: Static resource allocation for service differentiation and bandwidth diversity. In *Proceedings of IFIP WG 7.3 PERFORMANCE, Juan-Les-Pins, France*.

COHEN, B. 2003. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*.

FAN, B., CHIU, D.-M., AND LUI, J. 2006. Stochastic differential equation approach to model bittorrent-like file sharing systems. In *Proceedings of 14th IEEE International Workshop on Quality of Service, New Haven, CT, USA*.

FAN, B., LUI, J. C. S., AND CHIU, D.-M. 2009. The design trade-offs of bittorrent-like file sharing protocols. *IEEE/ACM Trans. Netw. 17,* 2, 365–376.

GAETA, R., GRIBAUDO, M., MANINI, D., AND SERENO, M. 2006. Analysis of resource transfers in peer-to-peer file sharing applications using fluid models. *Performance Evaluation 63,* 3, 149–174.

GUO, L., CHEN, S., XIAO, Z., TAN, E., DING, X., AND ZHANG, X. 2005. Measurements, analysis, and modeling of bittorrent-like systems. In *Proceedings of Internet Measurement Conference, Berkeley, CA, USA*.

GUO, L., CHEN, S., XIAO, Z., TAN, E., DING, X., AND ZHANG, X. 2007. A performance study of bittorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications 25,* 1, 155–169.

HALES, D. AND PATARIN, S. 2005. How to cheat bittorrent and why nobody does. Tech. Rep. UBLCS-2005-12, University of Bologna, Italy.

IZAL, M., KELLER, G., BIERSACK, E., FELBER, P., HAMRA, A., AND ERICE, L. 2004. Dissecting bittorrent: Five months in a torrent's lifetime. In *Proceedings of Passive and Active Measurements, Juan-les-Pins, France*.

LEGOUT, A., LIOGKAS, N., KOHLER, E., AND ZHANG, L. 2007. Clustering and sharing incentives in bittorrent systems. In *Proceedings of ACM SIGMETRICS, San Diego, California, USA*.

LIAO, W.-C., PAPADOPOULOS, F., AND PSOUNIS, K. 2006a. An efficient algorithm for resource sharing in peer-to-peer networks. In *Proceedings of IFIP-TC6 NETWORKING, Coimbra, Portugal*.

LIAO, W.-C., PAPADOPOULOS, F., AND PSOUNIS, K. 2006b. A peer-to-peer cooperation enhancement scheme and its performance analysis. *Journal of Communications 1,* 7, 24–35.

LIAO, W.-C., PAPADOPOULOS, F., AND PSOUNIS, K. 2007a. Performance analysis of bittorrent-like systems in heterogeneous networks. Tech. Rep. CENG-2007-2, University of Southern California.

LIAO, W.-C., PAPADOPOULOS, F., AND PSOUNIS, K. October 2007b. Performance analysis of bittorrent-like systems with heterogeneous users. In *Proceedings of IFIP WG 7.3 PERFORMANCE, Cologne, Germany*.

LIOGKAS, N., NELSON, R., KOHLER, E., AND ZHANG, L. 2006. Exploiting bittorrent for fun (but not profit). In *Proceedings of IPTPS, Santa Barbara, CA, USA*.

LO PICCOLO, F. AND NEGLIA, G. 2004. The effect of heterogeneous link capacities in bittorrent-like file sharing systems. In *Proceedings of Hot-P2P, Volendam, Nederlands*.

NEPHELAE. 2012. Nephelae Cloud Computing Cluster. `http://grid.ucy.ac.cy/Nephelae/`.

PARKER, A. 2004. The true picture of peer-to-peer filesharing. `http://www.cachelogic.com/`.

PIATEK, M., ISDAL, T., ANDERSON, T., KRISHNAMURTHY, A., AND VENKATARAMANI, A. 2007. Do Incentives Build Robustness in BitTorrent? In *Proceedings of NSDI, Cambridge, MA, USA*.

POUWELSE, J., GARBACKI, P., EPEMA, D., AND SIPS, H. 2005. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of IPTPS, Ithaca, NY, USA*.

QIU, D. AND SRIKANT, R. 2004. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM, Portland, OR, USA*.

RAO, A., LEGOUT, A., AND DABBOUS., W. 2010. Can Realistic BitTorrent Experiments Be Performed on Clusters? In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P), Delft, Netherlands*.

REN, S., TAN, E., LUO, T., CHEN, S., GUO, L., AND ZHANG, X. 2010. TopBT: a topology-aware and infrastructure-independent bittorrent client. In *Proceedings of IEEE INFOCOM, San Diego, California, USA*.

SAROIU, S., GUMMADI, K. P., DUNN, R. J., GRIBBLE, S., AND LEVY, H. M. 2002. An analysis of internet content delivery systems. In *Proceedings of OSDI, Boston, MA, USA*.

SHERMAN, A., NIEH, J., AND STEIN, C. 2009. Fairtorrent: bringing fairness to peer-to-peer systems. In *Proceedings of CoNEXT, Rome, Italy*.

SIRIVIANOS, M., PARK, J. H., CHEN, R., AND YANG, X. 2007. Free-riding in bittorrent networks with the large view exploit. In *Proceedings of IPTPS, WA, USA*.

THOMMES, R. AND COATES, M. December 2005. Bittorrent fairness: analysis and improvements. In *Proceedings of Workshop of Internet, Telecommunications and Signal Processing, Sunshine Coast, Australia*.

TIAN, Y., WU, D., AND NG, K. 2006. Modeling, analysis and improvement for bittorrent-like file sharing networks. In *Proceedings of IEEE INFOCOM, Barcelona, Spain*.

TORRENTFREAK. 2009. BitTorrent Still King of P2P Traffic. http://torrentfreak.com/bittorrent-still-king-of-p2p-traffic-090218/.

WANG, J., YEO, C., PRABHAKARAN, V., AND RAMCHANDRAN, K. 2007. On the role of helpers in peer-to-peer file download systems: Design, analysis, and simulation. In *Proceedings of IPTPS, WA, USA*.

Wolfram Mathematica. `http://www.wolfram.com/`.

YANG, X. AND VECIANA, G. D. 2004. Service capacity of peer to peer networks. In *Proceedings of IEEE INFOCOM, Hong Kong, China*.

Online Appendix to: Modelling BitTorrent-like systems with many classes of users

Wei-Cherng Liao, Netflame Technology Co., Ltd.
Fragkiskos Papadopoulos, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology
Konstantinos Psounis, Department of Electrical Engineering, University of Southern California
Constantinos Psomas, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology

A. STEADY STATE ANALYSIS FOR THE TOKEN-ENHANCED BITTORRENT SYSTEM

The model for the token-enhanced system is similar to the model for the original BitTorrent system. In particular, it is easy to see that Equations (1)...(9) in the main text hold for the token-enhanced system as well. Further, notice from Section 3 in the main text that the token-based scheme does not affect the rate by which a user uploads to a neighbor, since only neighbors with enough tokens to perform the download can be selected by a user. Therefore, it is not hard to justify why Equations (10)...(15) in the main text hold here as well.

Let's see what relations change compared to the original BitTorrent system. First, it is easy to see that in order to make the token-enhanced system operate properly we need to have $K_{\rm up} \geq K_{\rm down}$. Now, consider a L-BW leecher. Notice that $K_{\rm up}U_{i{\rm L}}$ $i \in \{{\rm H}, {\rm M}, {\rm L}\}$ is the token earning rate of an i-BW leecher from providing uploads to the L-BW leecher, and $K_{\rm down}D_{i{\rm L}}$ is the token spending rate of the i-BW leecher for downloading from the L-BW leecher. Since $K_{\rm up}U_{i{\rm L}} \geq K_{\rm down}U_{i{\rm L}} \geq K_{\rm down}U_{{\rm L}i} = K_{\rm down}D_{i{\rm L}}$ (as $K_{\rm up} \geq K_{\rm down}$ and $U_{i{\rm L}} \geq U_{{\rm L}i}$) for $i \in \{{\rm H}, {\rm M}, {\rm L}\}$, L-BW, M-BW, and H-BW leechers always have enough tokens to download from a L-BW leecher. (More generally, it is easy to see that a leecher with some upload link capacity will always have sufficient tokens to download from leechers with equal or lower upload link capacities.) Therefore, the L-BW leecher is equally likely to select a neighbor to provide uploads to, irrespectively of the neighbor's class. Since the total number of upload connections is Z, the percentage of H-BW and M-BW leechers in the system are $p_{\rm H}$ and $p_{\rm M}$ respectively, and the neighbor's list consists of a random selection of H-BW, M-BW, and L-BW leechers, in this system:

$$n^{\rm u}_{\rm LH} = Zp_{\rm H}. \tag{26}$$

$$n^{\rm u}_{\rm LM} = Zp_{\rm M}. \tag{27}$$

Now let's consider a M-BW leecher. The average number of L-BW neighbors that this M-BW leecher provides upload to, $n^{\rm u}_{\rm ML}$, is given by the following lemma:

LEMMA A.1.

$$n^{\rm u}_{\rm ML} = \min\left(Zp_{\rm L}, \frac{n^{\rm u}_{\rm LM}K_{\rm up}U_{\rm LM}p_{\rm L}}{K_{\rm down}U_{\rm ML}p_{\rm M}}\right). \tag{28}$$

PROOF. Both M-BW and H-BW leechers always have enough tokens to download from a M-BW leecher. Now, if L-BW leechers also always have enough tokens to download (i.e., $K_{\rm up}U_{\rm LM} \geq K_{\rm down}U_{\rm ML}$), then every leecher is equally likely to be selected for uploading to, irrespectively of its class, and thus $n_{\rm ML}$=$Zp_{\rm L}$. Otherwise, by observing that in the long run

© YYYY ACM 1049-3301/YYYY/01-ARTA $15.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

the token earning rate of all L-BW leechers from M-BW leechers ($n^{\mathrm{u}}_{\mathrm{LM}} K_{\mathrm{up}} U_{\mathrm{LM}} N p_{\mathrm{L}}$) equals the token spending rate of all L-BW leechers to M-BW leechers ($n^{\mathrm{u}}_{\mathrm{ML}} K_{\mathrm{down}} U_{\mathrm{ML}} N p_{\mathrm{M}}$), we have $n^{\mathrm{u}}_{\mathrm{ML}} = (n^{\mathrm{u}}_{\mathrm{LM}} K_{\mathrm{up}} U_{\mathrm{LM}} p_{\mathrm{L}})/(K_{\mathrm{down}} U_{\mathrm{ML}} p_{\mathrm{M}})$. □

Now, clearly, a M-BW leecher on average provides uploads to $(Z - n^{\mathrm{u}}_{\mathrm{ML}})$ M-BW and H-BW neighbors. Since both H-BW and M-BW leechers always have enough tokens to download from the M-BW leecher, it is easy to see:

$$n^{\mathrm{u}}_{\mathrm{MH}} = \frac{(Z - n^{\mathrm{u}}_{\mathrm{ML}}) p_{\mathrm{H}}}{p_{\mathrm{M}} + p_{\mathrm{H}}}. \tag{29}$$

Finally, using a similar reasoning as above, it is not hard to see that $n^{\mathrm{u}}_{\mathrm{HL}}$ and $n^{\mathrm{u}}_{\mathrm{HM}}$, are given as follows:

$$n^{\mathrm{u}}_{\mathrm{HL}} = \min\left(Z p_{\mathrm{L}}, \frac{n^{\mathrm{u}}_{\mathrm{LH}} K_{\mathrm{up}} U_{\mathrm{LH}} p_{\mathrm{L}}}{K_{\mathrm{down}} U_{\mathrm{HL}} p_{\mathrm{H}}}\right). \tag{30}$$

$$n^{\mathrm{u}}_{\mathrm{HM}} = \min\left(\frac{(Z - n^{\mathrm{u}}_{\mathrm{HL}}) p_{\mathrm{M}}}{p_{\mathrm{M}} + p_{\mathrm{H}}}, \frac{n^{\mathrm{u}}_{\mathrm{MH}} K_{\mathrm{up}} U_{\mathrm{MH}} p_{\mathrm{M}}}{K_{\mathrm{down}} U_{\mathrm{HM}} p_{\mathrm{H}}}\right). \tag{31}$$

From Equations (4)...(15) in the main text and (26)...(31), we can now compute n^{u}_{ij} and U_{ij} for all $i, j \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$, and then relate them to n^{d}_{ij} and D_{ij}, exactly as we did in the original BitTorrent system, in order to compute the user download rates.

B. SYSTEM TIME DYNAMICS IN THE TOKEN-ENHANCED BITTORRENT SYSTEM

It is easy to see that Equations (22)...(24) in the main text still hold in the token-enhanced system. What changes is the way we compute $\{\epsilon_{ji}(t)\}, i, j \in \mathbf{G}$. (Clearly the relation for $\epsilon_{si}(t)$ remains the same.)

Recall that a user earns K_{up} tokens for each byte he/she uploads and spends K_{down} tokens for each byte he/she downloads. Now, sort the user classes in accordance with their upload link capacities, with class 1 be the class with the lowest capacity. Along the same lines of the analysis in Section A, if leecher m belongs to class 1, we know that all of his/her neighbors always have sufficient tokens to download from him/her. Therefore, they equally share the upload link capacity of leecher m. This suggests that $\epsilon_{1i}(t) = x^{\mathrm{l}}_i(t)/\sum_{n=1}^{K} x^{\mathrm{l}}_n(t)$, $\forall i \in \mathbf{G}$. [1] Now, when a class 1 leecher exchanges data with a class 2 leecher, the token earning rate of the class 1 leecher from the class 2 leecher is $\epsilon_{12}(t)\mu_1 K_{\mathrm{up}}$, and the token spending rate of the class 1 leecher to the class 2 leecher is $\epsilon_{21}(t)\mu_2 K_{\mathrm{down}}$. Because in the long run the token earning rate of all class 1 leechers from class 2 leechers ($x^{\mathrm{l}}_1(t)\epsilon_{12}(t)\mu_1 K_{\mathrm{up}}$) equals the token spending rate of all class 1 leechers to class 2 leechers ($x^{\mathrm{l}}_2(t)\epsilon_{21}(t)\mu_2 K_{\mathrm{down}}$), we can write $\epsilon_{21}(t) = (x^{\mathrm{l}}_1(t)\epsilon_{12}(t)\mu_1 K_{\mathrm{up}})/(x^{\mathrm{l}}_2(t)\mu_2 K_{\mathrm{down}})$. However, notice that $\epsilon_{21}(t)$ cannot exceed the amount of the fair share that class 1 leechers can receive (from class 2 leechers) when class 1 leechers always have enough tokens to download. This amount is $x^{\mathrm{l}}_1(t)/\sum_{n=1}^{K} x^{\mathrm{l}}_n(t)$. Therefore:

$$\epsilon_{21}(t) = \min\left(\frac{x^{\mathrm{l}}_1(t)\epsilon_{12}(t)\mu_1 K_{\mathrm{up}}}{x^{\mathrm{l}}_2(t)\mu_2 K_{\mathrm{down}}}, \frac{x^{\mathrm{l}}_1(t)}{\sum_{n=1}^{K} x^{\mathrm{l}}_n(t)}\right). \tag{32}$$

[1]Notice that in contrast to Equation (25) in the main text this amount is independent of Z. This is because, a class 1 leecher in the token-based system on average uploads to $Z x^{\mathrm{l}}_i(t)/\sum_{n=1}^{K} x^{\mathrm{l}}_n(t)$ randomly selected class-i leechers ($i \geq 1$), as they all always have sufficient tokens. The fraction of the upload rate to each one of these is $1/Z$. To get $\epsilon_{1i}(t)$ we multiply these last two quantities, and Z cancels out. This holds for all $\epsilon_{ji}(t)$, whenever the class-i leechers have sufficient tokens to download from the class-j leechers, and it is the reason why Z does not appear in the subsequent equations.

Now, leechers in other classes share the remaining capacity of the class 2 leecher (because they all have sufficient tokens to exchange file blocks with this leecher, as they have larger upload link capacities, in accordance with our arguments in Section A). Hence:

$$\epsilon_{2i}(t) = \frac{(1-\epsilon_{21}(t))\, x_i^{\mathrm{l}}(t)}{\sum_{n=2}^{K} x_n^{\mathrm{l}}(t)}, \quad i \in \{2,\ldots,K\}.$$

Continuing this way, we can derive a general formula for $\epsilon_{ji}(t)$:

LEMMA B.1.

$$\epsilon_{ji}(t) = \begin{cases} \min\left(\frac{x_i^{\mathrm{l}}(t)\epsilon_{ij}(t)\mu_i K_{\mathrm{up}}}{x_j^{\mathrm{l}}(t)\mu_j K_{\mathrm{down}}}, \frac{x_i^{\mathrm{l}}(t)}{\sum_{n=1}^{K} x_n^{\mathrm{l}}(t)}\right) & \textit{if } i < j, \\ \frac{\left(1-\sum_{n=1}^{j-1}\epsilon_{jn}(t)\right)x_i^{\mathrm{l}}(t)}{\sum_{n=j}^{K} x_n^{\mathrm{l}}(t)} & \textit{otherwise.} \end{cases} \tag{33}$$

We can now solve the $2K$ differential equations as before, in order to study how the user population and download delays evolve over time in the token-enhanced system.

C. EXPERIMENTS FOR THE TOKEN-ENHANCED BITTORRENT SYSTEM

C.1. Steady State Performance Prediction and Flash Crowd Scenarios

Here we study the interesting dynamics of the token-based scheme. The simulation setup is the same as in Section 6.1 of the main text. We let $K_{\mathrm{down}} = 1$ and study how the system performs for different values of K_{up}. We also fix $L = 60$, which is a typical value in BitTorrent.

The download rates for all three classes of leechers are shown in Figure 10(i). From the plot we see again that theoretical and simulation results match. Further, we make the following interesting observation: The download rate of H-BW leechers first decreases and the download rate of L-BW leechers first increases, as K_{up} increases. This is because as K_{up} increases L-BW leechers earn tokens at a faster rate and they can download more data from H-BW leechers. This however means that H-BW leechers provide fewer uploads to other H-BW leechers. Thus, H-BW leechers have to download now from more L-BW leechers, and hence their download rate decreases. The download rate of M-BW leechers does not vary as much as K_{up} increases. This is explained as follows: The download rate of M-BW leechers from L-BW leechers does not change as K_{up} increases, since M-BW leechers always have enough tokens to download from L-BW leechers, i.e., even when $K_{\mathrm{up}} = 1$. As with L-BW leechers, as K_{up} increases, M-BW leechers can download more data from H-BW leechers, which could increase their download rate. However, at the same time, since L-BW leechers can download more data from M-BW leechers, it means that M-BW leechers provide fewer uploads to other M-BW leechers. Hence, the average download rate of M-BW leechers does not change much. Finally, note that the system throughput, i.e., the aggregate download rate of all leechers in the system, *does not* change as we vary K_{up}, and for large K_{up} each class of leechers has the same download rate. More precisely, this occurs when K_{up} is larger than the ratio of the largest upload link capacity to the smallest upload link capacity, times K_{down}, i.e., $K_{\mathrm{up}} > K_{\mathrm{down}}(C_{\mathrm{upH}}/C_{\mathrm{upL}}) = 6$ in our scenario, as all leechers, irrespectively of their class, always have enough tokens to initiate downloads and they cannot be distinguished by a potential uploader.

Figure 10(ii) shows the average file download delay of H-BW, M-BW, and L-BW leechers, and for the whole system. For comparison, the plot also shows the corresponding average download delay in the original BitTorrent system. As before, we observe that our model predicts the simulation results quite accurately. Further, we observe that when $K_{\mathrm{up}} = 1 = K_{\mathrm{down}}$, the performance of the token-enhanced system is almost identical to that of the original BitTorrent system. As K_{up} increases the overall delay improves. However, this occurs at the expense of the perceived delay of the H-BW leechers mostly, which increases.

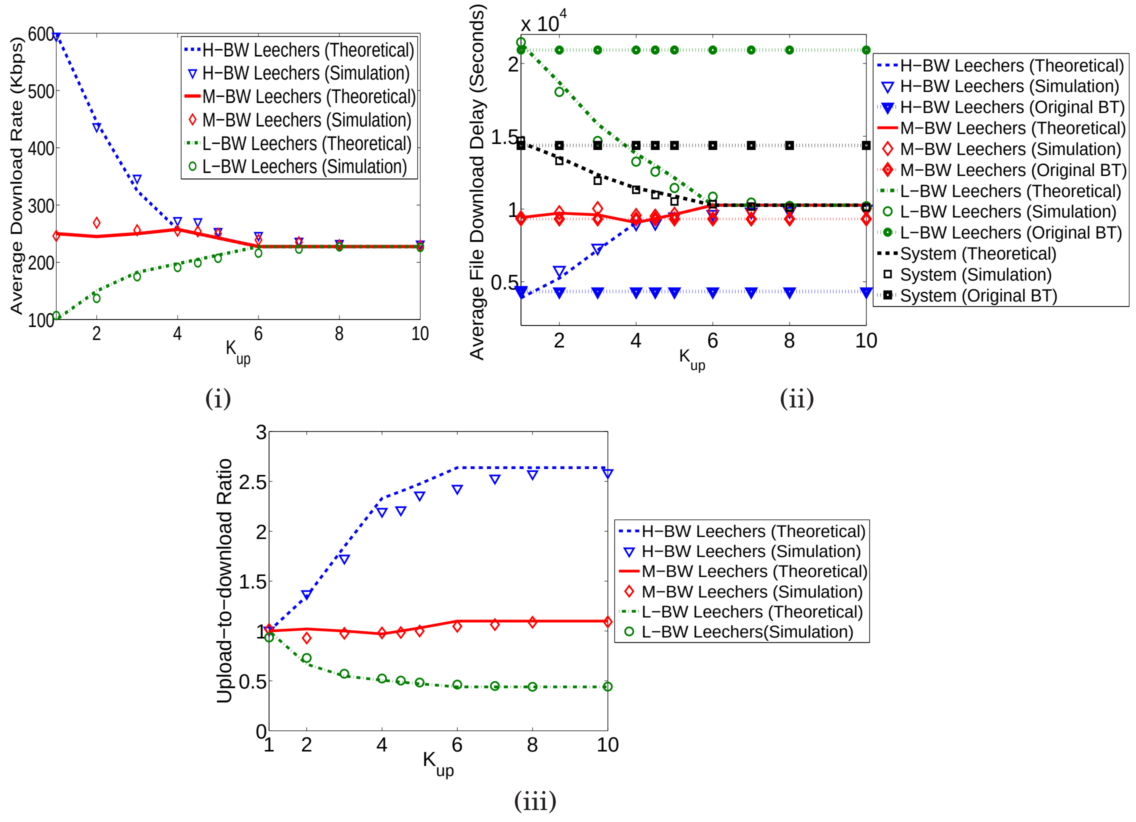


Fig. 10. The token-enhanced system: (i) Average download rate for H-BW, M-BW, and L-BW leechers, (ii) Average file download delay for H-BW, M-BW, and L-BW leechers, and for the system, (iii) Upload-to-download ratio for H-BW, M-BW, and L-BW leechers.

This motivates us to quantify next how "unfair" the token-based scheme becomes to H-BW leechers as K_{up} increases.

C.1.1. Overall Performance and Fairness. To quantify "fairness" we use the upload-to-download ratio of a user, which is defined as the user's upload rate divided by his/her download rate. This metric has been also used to quantify fairness in other studies as well, e.g., Bharambe et al. [2006], Thommes and Coates [2005]. Figure 10(iii) shows how the upload-to-download ratio behaves as we vary K_{up}, for H-BW, M-BW, and L-BW leechers.

From the plot we observe that the upload-to-download ratio is almost the same for all three classes of leechers when $K_{\text{up}} = 1 = K_{\text{down}}$. This implies that the system is fair. However, as K_{up} increases, the corresponding ratio for H-BW leechers increases and for L-BW decreases, as expected. This suggests that the system becomes unfair. For the M-BW leechers, this ratio remains almost invariant.

Looking at Figures 10(ii) and 10(iii) we can conclude that we can tradeoff between overall system performance and fairness to H-BW users. Using our analytical model we can predict how much "fairness" we are sacrificing and what performance is achieved. For example, one can enforce fairness by setting $K_{\text{up}} = 1 = K_{\text{down}}$, or can minimize the system's average download delay by choosing a large value for K_{up}. Further, one can also operate somewhere between these two extremes by setting the appropriate value for K_{up}.

Note that the issue of overall performance versus fairness is a controversial one. Our token-based scheme does not take sides, but instead provides a means to achieve any operational point.

C.2. Predicting System Time Dynamics and Non-flash Crowd Scenarios

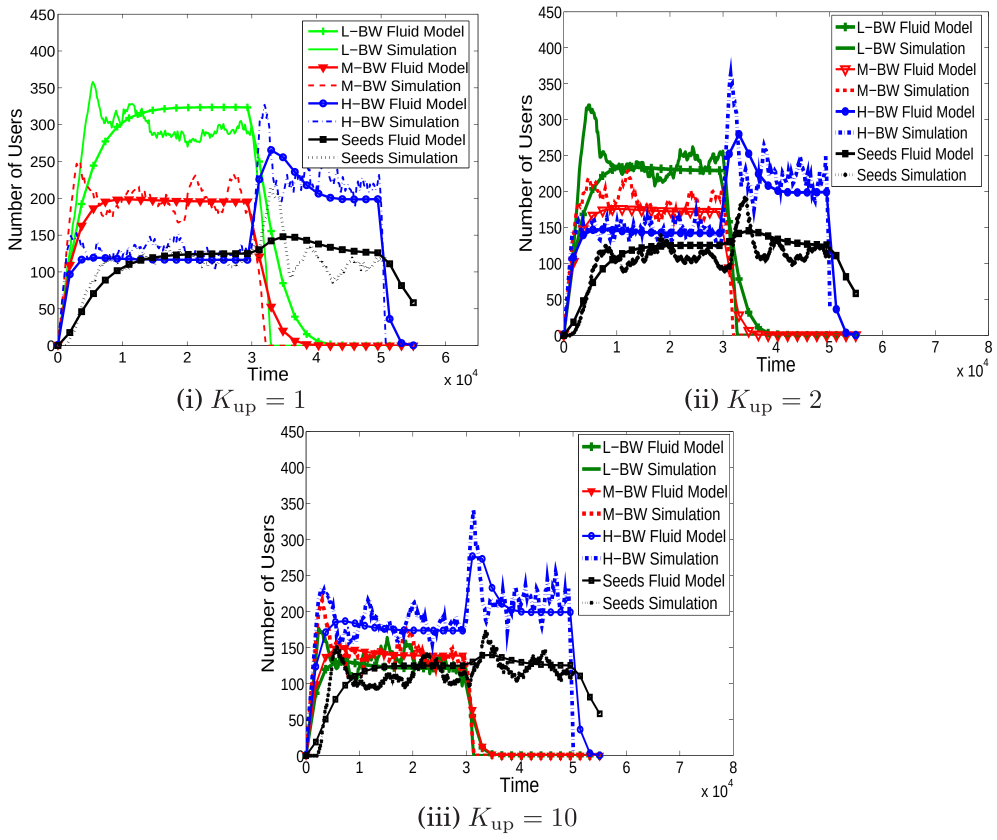


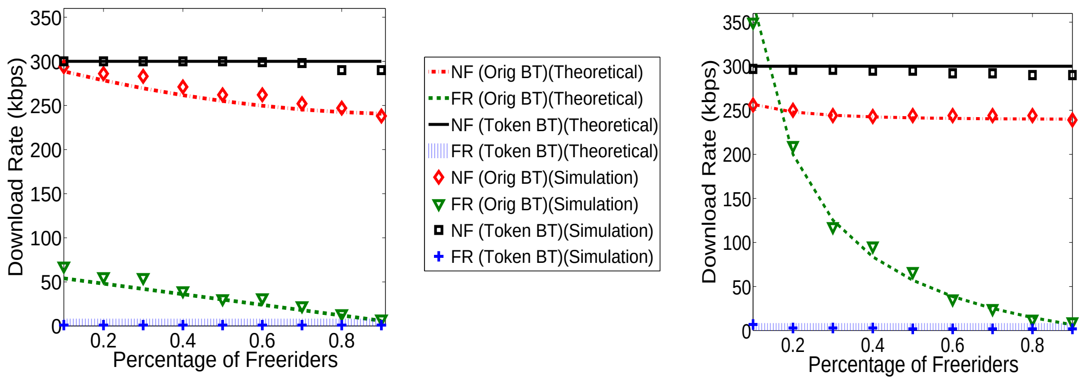
Fig. 11. The token-enhanced system: Number of users in the system.

The simulation setup is the same as in Section 6.2 of the main text. We fix $K_{\text{down}} = 1$ and vary K_{up} to study its effect on the system dynamics. We consider the arrival rate of leechers shown in Figure 4(i) of the main text. Figure 11 presents analytical and simulation results for the peer population in the token-enhanced system. In the figure we show results for $K_{\text{up}} = 1$, $K_{\text{up}} = 2$, and $K_{\text{up}} = 10$.

Again we observe that our model is quite accurate. We see that the token-enhanced system (Figure 11(i)) resembles the original BitTorrent system (Figure 4(ii) in the main text) when $K_{\text{up}} = 1 = K_{\text{down}}$, in agreement with our earlier arguments. The corresponding delay plot for $K_{\text{up}} = 1 = K_{\text{down}}$ is also similar to Figure 4(iii) in the main text. As we increase K_{up}, the number of lower bandwidth leechers in the system decreases and the number of higher bandwidth leechers increases. The explanation for this is the same as before. As K_{up} increases, lower capacity leechers can earn tokens at a faster rate, download faster, and hence depart the system earlier. On the other hand, higher bandwidth leechers download slower and hence depart the system after a longer period of time. Finally, when $K_{\text{up}} = 10$ the number of leechers present in the system from each class at steady state is proportional to the arrival rate of each class. This is explained as follows: As mentioned earlier, for large K_{up} ($K_{\text{up}} > 6$ in our scenario), all leechers can download at the same rate and hence the departure rate of each class of leechers is proportional to the steady state population of the class. Since the departure rate equals the arrival rate at steady state, the steady state population is proportional to the arrival rate. The download rate and delay of each class of users, as well as the fairness curves, behave, with respect to K_{up}, in a similar manner as in our steady state scenario (Figure 10).

C.3. Impact of the Proposed Token Based Scheme on Freeriders

As mentioned, the rate-based TFT scheme in general can motivate cooperation in BitTorrent. However, as reported in Liogkas et al. [2006], Hales and Patarin [2005] and Sirivianos et al. [2007], skillful freeriders can still benefit from the system by exploiting the optimistic unchoking scheme. In particular, they can connect to more peers than usual to increase the probability of receiving data via optimistic unchoking. In this section we study how the proposed token-based scheme prevents this type of freeriding in BitTorrent-like systems. To this end, we simulate both the original BitTorrent and the token-enhanced system (with $K_{\text{up}} = K_{\text{down}}$) with two classes of users: freeriders (FR) and non-freeriders (NF). We set the download link capacity of both classes of users to 10Mbps, the upload link capacity of non-freeriders to 300Kbps, and the upload link capacity of freeriders to 0Kbps since they do not offer any uploads. All other simulation parameters are the same as in Section 6.1 of the main text. We simulate two different scenarios. In the first scenario, both classes of users connect to $L = 40$ neighbors. In the second scenario, freeriders connect to all available leechers in an effort to maximally exploit optimistic unchoking.



(i) (ii)

Fig. 12. The token-based scheme prevents freeriding: (i) Scenario 1, and (ii) Scenario 2.

Figures 12(i) and 12(ii) show the download rates for freeriders and non-freeriders with respect to the percentage of freeriders in the system for Scenario 1 and 2 respectively. As before, theoretical results (from our first model) can predict simulation results well. Further, in Scenario 1 the download rate of freeriders is quite low under the original BitTorrent system, which implies that the system is working well. However, in Scenario 2 the download rate of freeriders is quite high and can be higher than even that of non-freeriders, when a small portion of skillful freeriders steal resource from a large portion of non-freeriders. This interesting observation has also been made in Sirivianos et al. [2007]. Nevertheless, the plots confirm our intuition that the token-based scheme does not suffer from this problem as it does not allow freeriders perform any downloads. This of course holds as long as K_{up} is not much larger than K_{down}. For K_{up} values significantly larger than K_{down}, freeriders may still be able to benefit from the system. For example, when $K_{\text{down}} = 1$ and $K_{\text{up}} = 10$, freeriders will need to upload only $1/10$ of a file to the system in order to retrieve from it the entire file. On the other hand, as we have seen earlier, larger K_{up} values also improve the overall performance of the system, compared to the original BitTorrent. Therefore, we see that a tradeoff exists again between overall system performance improvement and fairness.

D. PERFORMANCE PREDICTION FOR LARGE SYSTEMS

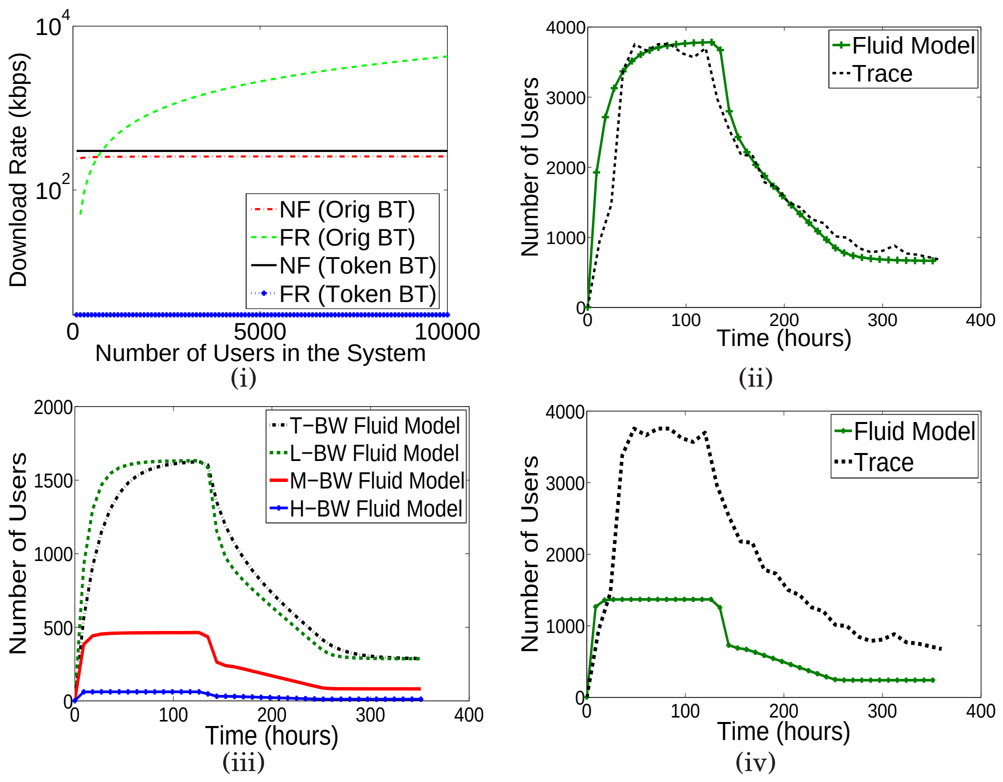


Fig. 13. Performance prediction for large systems: (i) The proposed token-based scheme can prevent freeriders from exploiting large systems, (ii) Our fluid model can reproduce the results of a real trace, (iii) The fluid model can further show the distribution of different classes of users, (iv) Using the proposed token-based scheme we can tradeoff fairness for overall system performance.

Having established the accuracy of the analytical models, in this section we use them to study the performance of large yet realistic BitTorent-like systems whose size makes simulation-based analysis too expensive.

The first system we consider consists of thousands of users. We use our first model to study this system and further argue that BitTorrent's TFT mechanism is not enough to deal with freeriders. We let 100 of those users to be freeriders, and the rest to be non-freeriders. The upload/download link capacity of freeriders and non-freeriders are the same as in the previous section. Figure 13(i) plots the download rate of freeriders as the total number of users in the system increases. Based on prior discussion, it is somewhat expected that freeriders will do really well as the number of non-freeriders increase, but still their performance is astonishing (the Y-axis on the plot is logarithmic). Essentially, freeriders connect to so many users that the aggregate download rate received via optimistic unchoking almost fills their download link capacity. As before, the token-based scheme solves this problem.

The second system we consider is based on a popular trace [Pouwelse et al. 2005] which measured the number of users downloading the *The Lord of The Rings* DVD. The trace spans a period of time that is a bit longer than two weeks, and reports on hundreds of thousands of users. The trace has no information about the type of users and their link capacities, so we use the trace-based studies in Saroiu et al. [2002] and Bharambe et al. [2006] to decide on the mix of users that we consider. Specifically, we assume four classes of users: H-BW (15%), M-BW (25%), L-BW (40%), and T-BW (20%) users (T-BW represents Tiny Bandwidth users) with capacities $C_{\text{upT}} = 128$Kbps, $C_{\text{downT}} = 784$Kbps, $C_{\text{upL}} = 384$Kbps, $C_{\text{downL}} = 1500$Kbps, $C_{\text{upM}} = 1000$Kbps, $C_{\text{downM}} = 3000$Kbps, and $C_{\text{upH}} = 5000$Kbps, $C_{\text{downH}} = 10000$Kbps.

We first find a set of arrival rates for each class of users that, when fed to our second, fluid-based model, yields a synthetic trace that is very similar to the original one, as shown in Figure 13(ii). We then use our model to predict the population of different classes of users in the system, as shown in Figure 13(iii). (No results of this type are available from the original trace.) Finally, we show what would have happened if the token-based scheme (with $K_{\text{up}} >> K_{\text{down}}$) had been used instead of the original TFT scheme. The average file download delay would have been significantly reduced, evident from the smaller number of users present in the system, at the expense of the perceived performance from higher bandwidth users as explained earlier.

Our motivation with these two examples has been to show the wide range of interesting results that one can obtain in no time using the set of equations that constitute our analytical models.

E. COMPARISON BETWEEN OUR TWO MODELS

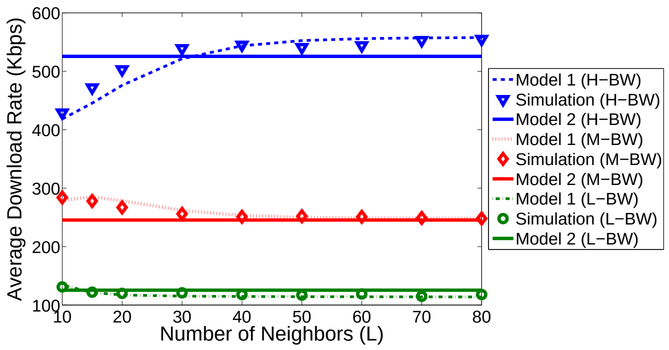


Fig. 14. Comparison between our two models.

As we have seen, our first model (presented in Section 4 of the main text) is tailored for steady state performance analysis, accounts for all central details of the BitTorrent protocol, and it is very accurate. It is now interesting to compare how accurate our fluid model is in such cases (i.e., for steady state performance analysis) and compare it with our first model. For this we consider the flash crowd scenario of Section 6.1 of the main text. We then solve our fluid model equations in steady state (i.e., in the interval $[t_1, t_2)$ in Figure 2 of the main text), and compute the corresponding leecher download rates. More precisely, we first set the number of class i leechers ($i \in \{\mathrm{H}, \mathrm{M}, \mathrm{L}\}$) to be $x_i^{\mathrm{l}}(t) = Np_i$, where Np_i is the number of class i leechers in steady state in the first model. (Recall from Section 6.1 in the main text that $N = 200, p_{\mathrm{L}} = 0.5, p_{\mathrm{M}} = 0.35, p_{\mathrm{H}} = 0.15$ and that there is also one seed in steady state.) Notice that for t in $[t_1, t_2)$ $x_i^{\mathrm{l}}(t)$ does not depend on t. Then, we substitute each $x_i^{\mathrm{l}}(t)$ in our fluid model with its steady state value Np_i, and compute the corresponding steady state values of $\epsilon_{ji}(t), \epsilon_{\mathrm{s}i}(t)$, and of the download rates $R_i(t)$ via Equation (22) in the main text.

The results are depicted in Figure 14. As we observe, our first model is more accurate as expected, since it captures the effect of connecting to a variable number of neighbors (L), and considers the optimistic unchoking reward scenario. Our second model is visibly inaccurate when L is small, because it does not account for leechers that don't have sufficient neighbors of the same class to connect to. However, the difference between the two models becomes small when L is sufficiently large, as expected. This difference now is only due to the fact that we do not consider the optimistic unchoking reward scenario in the second model. It is interesting to point out that this difference for H-BW leechers never exceeds $100/Z\%$. This is explained as follows. First recall that a H-BW leecher provides uploads to $Z - 1$ neighbors that provide him/her the highest download rates, and to one other neighbor via optimistic unchoking. The maximum rate the H-BW leecher can provide to the optimistic unchoked neighbor does not exceed $100/Z\%$ of its upload link capacity. When we consider the optimistic unchoking reward scenario, we are accounting for the fact that the optimistic unchoked neighbor will, in turn, reward the H-BW leecher with a download rate, which is at most equal to the one the optimistic unchoked neighbor receives from this leecher. Therefore, if we do not consider the optimistic unchoking reward scenario we may underestimate the H-BW leecher's download rate by at most $100/Z\%$. Finding a bound for the discrepancy between the two models for M-BW and L-BW leechers is much more involved, and depends on the values of several of the system parameters. We do not proceed with this task here. The interested reader is referred to Liao et al. [2007a].

As mentioned earlier, the first model requires $2n^2$ equations to model a system of n classes of users and does not model the system time dynamics. In contrast, the second model captures the system dynamics, and requires only $2n$ equations to model a system of n classes of users. In summary, the two models comprise a tradeoff between accuracy and simplicity/generality.