Performance Analysis of BitTorrent-like Systems in Heterogeneous Networks

Wei-Cherng Liao, Fragkiskos Papadopoulos, Konstantinos Psounis

E-mail: weicherl, fpapadop, kpsounis@usc.edu

Electrical Engineering Department

University of Southern California

Los Angeles, CA 90089

February 14, 2007

Performance Analysis of BitTorrent-like Systems in Heterogeneous Networks

Wei-Cherng Liao, Fragkiskos Papadopoulos, Konstantinos Psounis
Department of Electrical Engineering, University of Southern California, Los Angeles, USA
Email: {weicherl, fpapadop, kpsounis}@usc.edu

Abstract—**BitTorrent is the most successful peer-to-peer system and has attracted a lot of attention from the research community. 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 researchers to make a number of simplifying assumptions, e.g. user homogeneity, or even ignore some aspects of the protocol altogether, e.g. the 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 without compromising on the realism of the modeling methodology. 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 time-dynamics, we account for all the details of the BitTorrent protocol, including TFT and optimistic unchoking, and predict a number of performance metrics including upload and download rates, as well as file download delays. Our second model builds upon prior work on fluid models for BitTorrent. Using our first model, we extend the fluid-based methodology to capture the transient behavior as new users join or old users leave, while fully modelling central properties of the BitTorrent system, e.g. TFT. Finally, as an example of how to use our analytical models to study variations of the basic BitTorrent scheme and make design decisions, we propose a flexible token-based scheme for BitTorrent that can be used to tradeoff between overall system performance and fairness, and evaluate the scheme's parameters that achieve a target operational point.

Index Terms—**Heterogeneous P2P networks, BitTorrent, performance analysis, token based scheme, fairness/delay tradeoff.**

I. INTRODUCTION

Peer-to-peer (P2P) systems have provided a powerful infrastructure for large scale distributed applications, such as file sharing. As a result, they have become very popular. For example, 43% of the Internet traffic is P2P traffic [1]. Among all P2P systems, BitTorrent seems to be the most prevalent one. In particular, more than 50% of all P2P traffic is BitTorrent traffic [2].

The BitTorrent system is designed for efficient large scale content distribution. The complete BitTorrent protocol can be found in [3]. 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 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. An important one is the rate based TFT (Tit-for-Tat) unchoking scheme. In the rate based TFT unchoking scheme, a user will provide uploads to some of his/her neighbors (default is 4) who provide him/her the highest download rates and to one more, randomly selected neighbor, via a process called optimistic unchoking. This scheme discourages freeriders in the BitTorrent system because freeriders will keep getting choked if they do not provide uploads to other users.

Because of the prevalence and the success of BitTorrent, there is a large body of work that studies how it performs [4], [5], [6], [7], [8], [9], [10], [11], designs more incentive schemes for it [12], [13], collects traffic measurements [2], [14], [15], [16], and investigates fairness issues associated with it [17], [18], [19]. However, despite this large body of research, there have been very few attempts to mathematically model, in a heterogeneous and hence realistic environment, what is perhaps the most important performance metric from an end user's point of view: the average file download delay. Further, these attempts either make unnecessary simplifying assumptions, or completely ignore important aspects of the BitTorrent protocol that significantly affect performance.

In this paper we consider a heterogeneous BitTorrent-like system, where users may have different upload/download capacities, and propose two mathematical models that can accurately predict the user file download rates, and hence the download delays. Our first model is derived by considering the system performance in steady state, where the number of users in the system remains constant. This model accounts for *all* the details of the BitTorrent protocol and it is remarkably accurate. We demonstrate that it can be used to accurately predict the user file download delay for one of the most common scenarios in BitTorrent: the flash crowd scenario, where users join the system in a short time period just after a new file has been released [15], [17].

Although the aforementioned model is very detailed and accurate (for predicting performance in steady state), it does not capture the system's time dynamics (*e.g.*, such as the peer population evolution, etc.). Further, it becomes complicated when the network is "very heterogeneous", that is, when there is a large diversity in the upload/download capacities of users.

With the above in mind we propose a second model, which is derived along the same lines as the first model, but with some approximations that keep the analysis tractable. This second model comprises a *fluid model* that can predict not only the users' file download delays in a heterogeneous system, but also the system's time dynamics. This model is slightly less accurate than the first model (for steady state scenarios), however it is much simpler and general. Further, we demonstrate that it is quite accurate for predicting performance in non flash-crowd scenarios, where users will keep joining the system for a long time period after a file has been released.

Finally, we propose a token based TFT scheme, which is very simple and flexible. In the proposed scheme, which is inspired by our prior work on incentive schemes for P2P systems [20], [21], 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 by K_{up} for every byte he/she downloads from the neighbor. On the other hand, the user decreases a neighbor's tokens by K_{down} 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 proposed scheme can be used to tradeoff between high overall system performance and fairness to high bandwidth users, by properly setting its parameters K_{down} and K_{up}. In particular, we show that under the appropriate parameter tuning high bandwidth users will provide more uploads than usual to low bandwidth users, which tends to reduce the overall download delay. This however comes at the expense of making high bandwidth users download at a slower rate than they usually do. We extend our mathematical models to predict the average file download delays in this system, and demonstrate how the models can be used to decide on the values of K_{down} and K_{up} that achieve a target system performance/fairness.

The rest of this paper is organized as follows: In Section II we briefly discuss related work. In Section III we review the current BitTorrent implementation in more detail, and provide a detailed description of the proposed token based scheme. In Section IV we present our first mathematical model, which accurately predicts the performance of a heterogeneous BitTorrent-like system in steady state, and then extend this model to predict performance when the token based scheme is used. In Section V we present our second model, which predicts performance in a heterogeneous and dynamic BitTorrent-like system, with and without the token based scheme. In Section VI we present extensive simulation results in order to validate the accuracy of our models. In Section VII we compare our models with some of the most representative models in the literature. Conclusions and future work directions follow in Section VIII.

II. RELATED WORK

B. Cohen, the author of BitTorrent, gives a thorough introduction to the BitTorrent system in [22]. The paper describes the BitTorrent protocol, the system architecture and the incentive scheme built in the BitTorrent system. In addition,

there is a large body of work reporting the efficiency and the popularity of BitTorrent [11], [12], [15]. Although there are some studies, *e.g.* [12], [13], indicating that skillful freeriders can still benefit from the system against the built-in incentive scheme, BitTorrent in general has successfully motivated users to share their resources.

To our best knowledge, [4] is the first published work providing a mathematical model for the BitTorrent system. The paper proposes a fluid model to describe how the population of seeds and leechers evolves in the BitTorrent system. The studies in [6], [7], [9] extend the above model to study BitTorrent's performance under different user behaviors and different arrival processes. Further, [5] and [8] extend this model to study BitTorrent's performance under heterogeneous environments. In [10], the authors propose a model to study the peer distribution in BitTorrent and they use a dying process to study the file availability. Further, [23] uses a branching process to study the capacity of generic P2P systems, which have *some* similar characteristics to BitTorrent (partitioning of files into blocks, and multiple user upload connections), and the study in [24] proposes a fluid model to study such systems and verifies it numerically.

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 [4], [6], [7], [9] consider homogeneous network environments only, where users have the same link capacities. This is clearly an unrealistic assumption given Internet's heterogeneity. Further, while as we have mentioned above, the studies in [5] and [8] consider network heterogeneity, the study in [5] completely ignores BitTorrent's TFT scheme. And, the study in [8] attempts to model only some aspects of it under some simplifying assumptions. (See Section VII for these assumptions.) Further, these last two studies ([5] and [8]) are mostly theoretical and do not provide any simulation or experimental results to verify the validity of their model (instead, they only solve the derived equations numerically). However, as we have stressed earlier, BitTorrent's TFT scheme is one of the main features responsible for the system's great success.

Motivated by this, in this paper we propose two analytical models that accurately predict the performance of the BitTorrent system in both steady state and dynamic scenarios, under minimal assumptions, without compromising on the realism of the modeling methodology. In particular, we consider a *heterogeneous* (and hence realistic) BitTorrent-like system, where users are grouped into different classes according to their link capacities, and attempt to *fully* model many important aspects of the system, *including* its TFT scheme. Interestingly enough, despite the protocol's complexity, we show that it is possible to accomplish this, even if the network environment consists of an arbitrarily large number of user classes, while keeping the analysis simple and the performance prediction accurate.

The work in [17] proposes a block based TFT scheme. Our proposed token based TFT scheme, which is inspired by our prior work on incentive schemes for P2P systems [20], [21], degenerates to the scheme in [17] when $K_{up} = K_{down}$. Hence, our scheme is much more general and flexible. Further, the

work in [17] studies the performance of the block based TFT scheme only via simulations. Here, we extend our mathematical models to predict the performance of the token based scheme for a general $\frac{K_{up}}{K_{down}}$ ratio. Finally, we show how our models can be used to decide on the scheme parameters that achieve a target tradeoff between overall system performance and fairness to high bandwidth users.

III. PRELIMINARIES

A. *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 $40 - 60$ [3].) 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 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 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 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, $X = 4$.) Similarly, 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. (Therefore, by default $Z = 5$.) 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. (Clearly, $t_2 \geq t_1 + 10$ seconds). 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 $t_2 < t_1 + 30$ seconds, the user does not choke the optimistic unchoking neighbor and does not perform a new optimistic unchoking, and (ii) if $t_2 \geq t_1 + 30$ seconds, the user chokes the optimistic unchoking neighbor and performs a new optimistic unchoking. We call this 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.

B. *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 $K_{down}X_{up}$. On the other hand, the user increases a neighbor's tokens by $K_{up}X_{down}$ if he/she downloads X_{down} bytes from the neighbor under study. 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. This is equal to 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_{up} and K_{down} are relative values. Therefore, the proposed scheme actually has only one design parameter. We will show that for $K_{up} = K_{down}$ the token-based system has approximately the same performance, and it is as fair, as the original BitTorrent system. Finally, we will also show that as K_{up} increases the overall system performance of the token-based system can get significantly better than that of the original BitTorrent system, by sacrificing some fairness

towards high capacity users. In particular, high capacity users will end up providing uploads to the system at a faster rate than the download rate they receive.

IV. STEADY STATE ANALYSIS

In this section we propose a mathematical model to study the performance of a BitTorrent-like system in steady state, where the number of leechers and seeds in the system is assumed to remain constant over a relatively long period of time. Such an assumption is not unrealistic in flash crowd scenarios [15], [17]. (However, note that we will relax it in the next section.)

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. As mentioned earlier, in real P2P systems users have heterogeneous capacities. We incorporate this fact in our analysis in order to make it more realistic and general. Further, we also consider all details of the BitTorrent protocol. For ease of exposition, assume that there exist two classes of users (leechers): (i) high bandwidth (**H-BW**) users, who have a high upload link capacity, and (ii) low bandwidth (**L-BW**) users, who have a low upload link capacity. We will show in IV-D that the model can be extended along the same lines for more classes of users, in accordance with recent trace-based studies [17], [25] that divide the users of a real P2P system into four classes. (Also, in the next section we present a second model that can easily account for arbitrarily many classes of users, however by sacrificing some accuracy.) We denote by N the total number of leechers in the system and by α the percentage of L-BW leechers. We start our analysis with the original BitTorrent system and then proceed with the token-enhanced system.

A. *Computing the User Download Rates in the Original BitTorrent System*

Consider a H-BW leecher and denote by n_{HH}^d and n_{HL}^d the steady state average number of H-BW and L-BW neighbors respectively that this leecher is downloading from, and by D_{HH} and D_{HL} the corresponding average download rates. Similarly, consider a L-BW leecher and denote by n_{LH}^d and n_{LL}^d the steady state average number of H-BW and L-BW neighbors respectively that this leecher is downloading from, and by D_{LH} and D_{LL} the corresponding average download rates. Further, let D_S be the average download rate that a leecher can receive from the seed(s). Now, let R_{downH} and R_{downL} be the aggregate download rate of a H-BW and a L-BW leecher respectively. It is easy to see that:

$$R_{downH} = n_{HH}^d D_{HH} + n_{HL}^d D_{HL} + D_S, \quad (1)$$
$$R_{downL} = n_{LH}^d D_{LH} + n_{LL}^d D_{LL} + D_S. \quad (2)$$

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

Now, denote by n_{HH}^u and n_{HL}^u the steady state average number of H-BW and L-BW neighbors respectively that a H-BW leecher is uploading to, and let U_{HH} and U_{HL} be the corresponding average upload rates. Similarly, denote by n_{LH}^u and n_{LL}^u the steady state average number of H-BW and L-BW neighbors respectively that a L-BW leecher is uploading to, and by U_{LH} and U_{LL} the corresponding average upload rates. Further, let R_{upH} and R_{upL} be the aggregate upload rate of a H-BW and a L-BW leecher respectively. As before, we can write:

$$R_{upH} = n_{HH}^u U_{HH} + n_{HL}^u U_{HL}, \quad (3)$$
$$R_{upL} = n_{LH}^u U_{LH} + n_{LL}^u U_{LL}. \quad (4)$$

In order to be able to predict the download delays we first need to compute the download rates R_{downH} and R_{downL}. Hence, we need to calculate the values of the parameters n_{HH}^d, n_{HL}^d, n_{LH}^d, n_{LL}^d, D_{HH}, D_{HL}, D_{LH}, and D_{LL}. To do so, we first compute the values of n_{HH}^u, n_{HL}^u, n_{LH}^u, n_{LL}^u, U_{HH}, U_{HL}, U_{LH}, and U_{LL} (that comprise Equations (3) and (4)) and then relate them to the aforementioned parameters. Notice that computing these parameters first is easier. This is because, it is the rules according to which a user chooses a neighbor to provide *uploads to*, that are explicitly defined in the BitTorrent protocol. However, in order to compute them we first need to find, in addition to Equations (3) and (4), six more relations. In this way we will have a system comprising of eight equations and eight unknowns. [1] For this, we proceed as follows.

Let Z be the number of neighbors that a user in BitTorrent is uploading to at any time instance. (This is a system configuration parameter; by default $Z = 5$.) Hence:

$$n_{HH}^u + n_{HL}^u = Z, \quad (5)$$
$$n_{LH}^u + n_{LL}^u = Z. \quad (6)$$

Now let C_{upH}/C_{downH} and C_{upL}/C_{downL} be the upload/download link capacity of H-BW and L-BW leechers respectively. Further, assume that a leecher's download link capacity is larger than or equal to his/her upload link capacity. Therefore, the system's bottlenecks are the upload links and we can assume that these are fully utilized. [2] This means that $R_{upH} = C_{upH}$ and that $R_{upL} = C_{upL}$.

Since peer-to-peer traffic is transferred via TCP connections, we can further assume that the upload capacity of a user will be fairly shared among concurrent upload connections, *if* the maximum possible download rate of *each* connection is larger or equal to the fair share. (This is a working assumption, also made in many other studies of P2P systems, *e.g.* see [26].) For L-BW leechers this is always the case since $C_{downH} > C_{upL}$,

---

[1] 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.

[2] This is not an unrealistic assumption. Common Internet access technologies, such as Dial-up, DSL, cable-modem, and Ethernet, satisfy this assumption [25]. Further, this assumption has been also made in many other studies on peer-to-peer networks, *e.g.* see [9] and references therein. And, it is in accordance with measurement studies of BitTorrent systems, *e.g.* see [11], [15].

and $C_{downL} \geq C_{upL}$, and hence we can write the following equation:

$$U_{LL} = U_{LH} = \frac{C_{upL}}{Z}. \qquad (7)$$

We now turn our attention to the upload rate that a H-BW leecher provides to a L-BW leecher (U_{HL}). At any time instance a L-BW leecher is downloading on average from n_{LL}^d L-BW neighbors. We define the *spare* download capacity of this leecher as $C_{downL} - n_{LL}^d D_{LL} - D_S$. Then, the upload rate that a H-BW leecher can provide to a L-BW leecher is given by the following lemma: [3]

Lemma 1:

$$U_{HL} = \min\left(\frac{C_{upH}}{Z}, C_{downL} - n_{LL}^u U_{LL} - D_S\right). \qquad (8)$$

Proof: If the spare capacity of the L-BW leecher is larger than his fair share ($\frac{C_{upH}}{Z}$), the leecher will be downloading from the H-BW leecher at an average rate equal to his/her fair share. Otherwise, the leecher will be downloading at an average rate equal to his/her spare capacity ($C_{downL} - n_{LL}^d D_{LL} - D_S$). However, notice that $n_{LL}^d D_{LL} = n_{LL}^u U_{LL}$, as the total download rate from L-BW leechers to L-BW leechers equals the total upload rate from L-BW leechers to L-BW leechers. ∎

Now, note that once we have an expression for one of n_{HH}^u or n_{HL}^u (and hence for the other by Equation (5)), the expression for U_{HH} will result from Equation (3). We proceed with n_{HL}^u (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. [4] Further, denote by $Binomial(N, p, k)$ the probability mass function of a Binomial random variable with parameters N and p, that is, $Binomial(N, p, k) \equiv \binom{N}{k} p^k (1-p)^{(N-k)}$. Then, n_{HL}^u is given by the following lemma:

Lemma 2:

$$n_{HL}^u = \sum_{k=0}^{L} n(k) P\{\text{have } k \text{ H-BW neighbors out of } L\}, \qquad (9)$$

where:

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

and:

$$P\{\text{have } k \text{ H-BW neighbors out of L}\} = Binomial(L, 1-\alpha, k).$$

Proof: First recall that α is the percentage of L-BW leechers in the system. Since the neighbors' list consists of a random selection of H-BW and L-BW leechers, it is easy to see that $P\{\text{have } k \text{ H-BW neighbors out of L}\} = Binomial(L, 1-\alpha, k)$. Now let's consider a H-BW leecher,

---

[3]Note that because of BitTorrent's TFT strategy (see Section III), the probability that the *same* L-BW leecher is concurrently downloading from two or more H-BW leechers is quite small.

[4]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 [17]. This is partially due to the local rarest first (LRF) block selection algorithm that BitTorrent uses to disseminate blocks.

---

say leecher j, and let $k \leq L$ be the number of j's H-BW neighbors. Since j provides uploads to Z of his/her neighbors, we distinguish two cases: (i) $k \geq Z$, and (ii) $k < Z$. First, consider case (i) and recall how BitTorrent's TFT scheme works (see Section III). It is easy to see that in this case j 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 $\frac{L-k}{L-Z+1}$. Now consider case (ii). In this case j is uploading to exactly $Z - k$ L-BW leechers at any time instance, as he/she does not have any other H-BW neighbor that he/she could provide uploads to. It is now easy to see that n_{HL}^u is given by Equation (9). ∎

Finally, we also need to find an expression for one of n_{LH}^u or n_{LL}^u. (The other will result from Equation (6)). To compute n_{LH}^u (the average number of H-BW leechers that a L-BW leecher provides uploads to), we proceed as follows. First, recall from Section III that the optimistic unchoking period is 30 seconds, the rate observation window is 20 seconds, and users make their choking decision every 10 seconds. Suppose that H-BW leecher j selects 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.
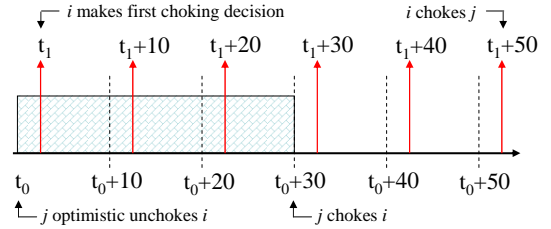

Fig. 1. Time line of optimistic unchoking and choking decision making.

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). [5] 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 $\frac{U_{HL}(20+t_0-t_1)}{20}$. 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 $\frac{U_{HL}(10+t_0-t_1)}{20}$. If this rate is larger than U_{LL}, i will not choke j. Therefore, if $N_{unchoke}$ denotes the number of times that i did not choke j in the time-interval $[t_1, t_1 + 50]$, we can write:

$$N_{unchoke} = \begin{cases} 3 & \text{if } U_{HL}\frac{(20+t_0-t_1)}{20} < U_{LL}, \\ 5 & \text{if } U_{HL}\frac{(10+t_0-t_1)}{20} \geq U_{LL}, \\ 4 & \text{otherwise.} \end{cases}$$

---

[5]Recall that the probability that two or more H-BW leechers uploading to the same L-BW leecher at the same time-instance is small. Therefore, i will be always downloading from at least one L-BW neighbor.

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

Recall that a H-BW leecher is uploading to n_{HL}^u 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 $n_{HL}^u \frac{10 \overline{N}_{unchoke}}{30}$ 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 $\left(\frac{1-\alpha}{\alpha} \right) n_{HL}^u \frac{10 \overline{N}_{unchoke}}{30}$. We refer to this scenario, as the *optimistic unchoking reward scenario*.

n_{LH}^u is now given by the following lemma:

Lemma 3:

$$n_{LH}^u = \sum_{w=0}^{L} n(w) P_1(w) + \left(\frac{1-\alpha}{\alpha} \right) n_{HL}^u \frac{\overline{N}_{unchoke}}{3}, \quad (10)$$

where:

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

and:

$$
\begin{aligned}
P_1(w) &= P\{\text{have } w \text{ L-BW neighbors out of L}\} \\
&= Binomial(L, \alpha, w).
\end{aligned}
$$

Proof: As before, since α is the percentage of L-BW users in the system and the neighbors' list consists of a random selection of H-BW and L-BW users, the probability of having w L-BW neighbors out of L is $Binomial(L, \alpha, w)$. Further, the second term on the right hand side of Equation (10) corresponds to the optimistic unchoking reward scenario. What about the first term? This term accounts for the number of H-BW users that a L-BW user has chosen to upload to, just like in the proof of Lemma 2.

In particular, consider L-BW user i, and let $w \leq L$ be the number of i's L-BW neighbors. As before, we distinguish two cases: (i) $w \geq Z$, and (ii) $w < Z$. In case (i) i may be uploading to at most one H-BW user at any time instance. This H-BW user has been selected via optimistic unchoking, with probability $\frac{L-w}{L-Z+1}$, and will be choked after the optimistic unchoking period elapses. This is because the H-BW user, who prefers other H-BW users to upload to, won't be uploading to this L-BW user. In case (ii), i has selected to upload to exactly $Z - w$ H-BW users, as he/she does not have any other L-BW neighbor to provide uploads to. ∎

Notice that in Lemma 2 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.)

Given Equations (3)...(10) (and the fact that $R_{upH} = C_{upH}$, $R_{upL} = C_{upL}$), we can now compute n_{HH}^u, n_{HL}^u, n_{LH}^u, n_{LL}^u, U_{HH}, U_{HL}, U_{LH}, and U_{LL}. We proceed to relate these parameters to n_{HH}^d, n_{HL}^d, n_{LH}^d, n_{LL}^d, D_{HH}, D_{HL}, D_{LH}, and D_{LL}. First, clearly $D_{HH} = U_{HH}$, $D_{HL} = U_{LH}$, $D_{LH} = U_{HL}$, and $D_{LL} = U_{LL}$. Further, notice that 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. Therefore, we can write $n_{LH}^d \alpha = n_{HL}^u (1 - \alpha)$. Similarly, we can easily relate n_{HH}^d, n_{HL}^d, n_{LL}^d to n_{HH}^u, n_{LH}^u, n_{LL}^u, as follows: $n_{HH}^d = n_{HH}^u$, $n_{HL}^d (1 - \alpha) = n_{LH}^u \alpha$, and $n_{LL}^d = n_{LL}^u$. Hence, we can now compute the average download rate of a H-BW leecher and a L-BW leecher using Equations (1) and (2), and of course the average download rate across all users.

B. Computing the User Download Rates in 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)...(6) hold for the token-enhanced system as well.

As before, we assume again that the download capacity of a user is larger than or equal to his/her upload capacity. Now, 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. For a L-BW leecher, his/her L-BW neighbors may earn tokens by uploading to him/her at a rate $K_{up} U_{LL}$, and they spend tokens by downloading from him/her at a rate $K_{down} D_{LL}$. Clearly, to make the token based system operate properly, we need to have $K_{up} \geq K_{down}$. Hence, $K_{up} U_{LL} \geq K_{down} D_{LL}$ (since $D_{LL} = U_{LL}$). Now, consider a H-BW leecher. The rate that a H-BW leecher gains tokens by providing uploads to a L-BW leecher ($K_{up} U_{HL}$) is larger than the rate that the leecher spends tokens by downloading from the L-BW leecher ($K_{down} D_{HL}$), since $K_{up} U_{HL} \geq K_{down} U_{HL} > K_{down} U_{LH} = K_{down} D_{HL}$. Therefore, all users will always have enough tokens to download from a L-BW leecher. Hence, the upload capacity of a L-BW leecher is fully utilized and Equation (7) holds true in this system as well.

Now, let's see what relations change compared to the original BitTorrent system. Recall that according to the token based scheme a user randomly selects to upload to those neighbors who have enough tokens to perform the download. Consider a L-BW leecher. Since $K_{up} U_{LL} \geq K_{down} D_{LL}$ (as $K_{up} \geq K_{down}$ and $U_{LL} = D_{LL}$), and $K_{up} U_{HL} \geq K_{down} U_{HL} > K_{down} U_{LH} = K_{down} D_{HL}$ (as $K_{up} \geq K_{down}$, $U_{HL} > U_{LH}$, and $U_{LH} = D_{HL}$), both L-BW and H-BW leechers always have enough tokens to download from a L-BW leecher. Therefore, the L-BW leecher will equally select every peer to provide uploads to. Since the total number of upload connections is Z, the percentage of H-BW leechers in the system is $1 - \alpha$, and the neighbor's list consists of a random selection of H-BW and L-BW leechers, in this system:

$$n_{LH}^u = Z(1 - \alpha). \quad (11)$$

Now let's consider a H-BW leecher and first concentrate on the the rate by which this leecher provides uploads to a L-BW leecher. First assume that $K_{up}U_{LH} \geq K_{down}U_{HL}$. Under this condition a L-BW leecher earns tokens by uploading to a H-BW leecher at a faster rate than the rate that he/she spends tokens by downloading from the H-BW leecher. This means that a L-BW leecher always has enough tokens to download from a H-BW leecher. Since a H-BW leecher always has enough tokens to download from a H-BW leecher as well (since $K_{up}U_{HH} \geq K_{down}D_{HH}$, as $K_{up} \geq K_{down}$ and $U_{HH} = D_{HH}$), the H-BW leecher cannot distinguish H-BW neighbors from L-BW neighbors, and thus he/she provides uploads to all of his/her neighbors with the same probability. Hence:

$$n_{HL}^u = Z\alpha. \tag{12}$$

Further, U_{HL} in this scenario is given in the following lemma:

Lemma 4:

$$U_{HL} = \sum_{i=0}^{L} \sum_{k=0}^{i} \min\left(\frac{C_{upH}}{Z}, R_{HL}(k)\right) P_1(k|i) P_2(i), \tag{13}$$

where:

$$R_{HL}(k) = \begin{cases} \frac{C_{downL} - n_{LL}^d U_{LL} - D_S}{k} & \text{if } k > 0, \\ 0 & \text{otherwise.} \end{cases}$$

and:

$$P_1(k|i) = P\{\text{download from } k \text{ out of } i \text{ H-BW neighbors}\}$$
$$= Binomial(i, \frac{Z}{L}, k),$$
$$P_2(i) = P\{\text{have } i \text{ H-BW neighbors out of L}\}$$
$$= Binomial(L, 1-\alpha, i).$$

Proof: First, as we have said, now a L-BW user always has enough tokens to download from a H-BW neighbor. Hence, her/her download rate is not constrained by the amount of tokens he/she possesses. If a L-BW user is downloading from $k > 0$ H-BW users, the average download rate from each H-BW user is equal to $R_{HL}(k) = \frac{C_{downL} - n_{LL}^d U_{LL} - D_S}{k}$, where $C_{downL} - n_{LL}^d U_{LL}$ is the spare capacity of the L-BW user. However, this rate cannot exceed the maximum average rate that a L-BW user can download from a H-BW user, which is $\frac{C_{upH}}{Z}$. Further, the probability that the L-BW user is downloading from a H-BW neighbor is $\frac{Z}{L}$ because each user randomly selects Z out of L neighbors to provide uploads to (as every neighbor always has enough tokens). Therefore, given that a L-BW user has i H-BW neighbors, the probability that he/she is downloading from $k \leq i$ of them is $Binomial(i, \frac{Z}{L}, k)$. And finally, the probability that the L-BW user has i H-BW neighbors is $Binomial(L, 1-\alpha, i)$. ∎

Notice that under the aforementioned condition ($K_{up}U_{LH} \geq K_{down}U_{HL}$) the upload link capacity of a H-BW leecher may not be fully utilized. This is because, since every neighbor seems identical, a H-BW leecher may select to provide uploads to *several* L-BW leechers who cannot download fast. Hence, it is no longer the case that $R_{upH} = C_{upH}$, and therefore, we cannot use Equations (3),

(5), (12) and (13) to compute U_{HH}. Instead, U_{HH} is given below:

Lemma 5:

$$U_{HH} = \sum_{w=0}^{Z} R_{HH}(w) P\{\text{upload to } w \text{ L-BW neighbors}\}, \tag{14}$$

where:

$$R_{HH}(w) = \begin{cases} \frac{C_{upH} - wU_{HL} - D_S}{Z - w} & \text{if } w < Z, \\ 0 & \text{otherwise.} \end{cases}$$

and:

$$P\{\text{upload to } w \text{ L-BW neighbors}\} = Binomial(Z, \alpha, w).$$

Proof: The average rate by which a H-BW leecher is uploading to a L-BW leecher is U_{HL}. If a H-BW leecher is uploading to w L-BW leechers, then the average upload rate to each H-BW leecher is equal to $R_{HH}(w) = \frac{C_{upH} - wU_{HL} - D_S}{Z - w}$, where $C_{upH} - wU_{HL} - D_S$ is the spare upload capacity of the H-BW leecher. Further, a H-BW leecher randomly selects Z neighbors to provide uploads to because they always have tokens. Hence, $P\{\text{upload to } w \text{ L-BW neighbors}\} = Binomial(Z, \alpha, w)$. ∎

From Equations (3)...(7) and (11)...(14) (as well as the fact that $R_{upL} = C_{upL}$), we can now compute n_{HH}^u, n_{HL}^u, n_{LH}^u, n_{LL}^u, U_{HH}, U_{HL}, U_{LH} and U_{LL}. And, we can relate these parameters to n_{HH}^d, n_{HL}^d, n_{LH}^d, n_{LL}^d, D_{HH}, D_{HL}, D_{LH} and D_{LL}, exactly as we did in the original BitTorrent system. Therefore, we can now compute the average download rates using Equations (1) and (2).

Note that in order to check whether condition $K_{up}U_{LH} \geq K_{down}U_{HL}$ is satisfied, after solving the system of equations under this assumption, as described above, we then need to check if the resulting U_{HL} satisfies this condition ($K_{up}U_{LH} \geq K_{down}U_{HL}$). If the condition is not satisfied Equations (12)...(14) do not hold. This means that we need to find new expressions, and resolve a system of Equations with these new expressions. This is because, if the condition is not satisfied, it means that L-BW leechers will not have sufficient tokens to download from H-BW leechers, and therefore, H-BW leechers will rarely pick L-BW leechers to upload to. The resulting relation for U_{HL} in this case is similar to Equation (8) and it is given in the following lemma:

Lemma 6:

$$U_{HL} = \min\left(\frac{C_{upH}}{Z}, C_{downL} - n_{LL}^d U_{LL} - D_S, \frac{K_{up}U_{LH}}{K_{down}}\right). \tag{15}$$

Proof: First, the token earning rate of a L-BW user from a H-BW user is $K_{up}U_{LH}$. Hence, the download rate of a L-BW user from a H-BW user cannot exceed $\frac{K_{up}U_{LH}}{K_{down}}$. (Recall that each user keeps track of the amount of tokens that his/her neighbor possesses.) Now, $C_{downL} - n_{LL}^d U_{LL} - D_S$ is the spare download capacity of the L-BW user. Clearly, he/she cannot download at a rate faster than this. Finally, as with the proof of Lemma 1, if the spare capacity of the L-BW user is larger than his/her fair share ($\frac{C_{upH}}{Z}$), the user will be downloading from the H-BW user at an average rate equal to his/her fair share. Combining these facts, gives the result. ∎

For n_{HL}^u, by observing that in the long run the token earning rate of all L-BW leechers from H-BW leechers ($n_{LH}^u K_{up} U_{LH} N \alpha$) equals the token spending rate of all L-BW leechers to H-BW leechers ($n_{HL}^u K_{down} U_{HL} N (1-\alpha)$), we can write:

$$n_{HL}^u = \frac{n_{LH}^u K_{up} U_{LH} \alpha}{K_{down} U_{HL}(1-\alpha)}, \qquad (16)$$

where U_{LH} is given as before by Equation (7), n_{LH}^u by Equation (11), and U_{HL} by Equation (15).

Finally, since H-BW leechers will rarely pick L-BW leechers to upload to, we can now use the fact that $R_{upH} = C_{upH}$, and therefore, we do not have to find an explicit formula for U_{HH}. As with the rest of the unknowns, its value will result by solving the system comprising of Equations (3)...(7) and (11), (15), (16) (using the facts that $R_{upL} = C_{upL}$ and $R_{upH} = C_{upH}$). And, we can now compute the download rates as before.

C. *Estimating the Average Download Delay*

So far, we have seen how one can compute the download rates in a heterogeneous BitTorrent-like system with and without the token based scheme, 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 [15], [17]. 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 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, both H-BW leechers and L-BW leechers are present in the system. Since H-BW leechers have higher capacities, they depart earlier, by time t_3. Afterward, 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 download rate of L-BW leechers during the interval $(t_3, t_4]$ is just equal to the sum of their upload link capacity, since this is fully utilized as explained earlier, and the download rate they receive from seeds. Notice that our earlier analysis does not model the transient periods $(t_0, t_1]$ and $(t_2, t_3]$. To compute the delays we make the assumption that $t_0 \approx t_1$ and $t_2 \approx t_3$. [6]

Now, let S be the file size and let T_H and T_L be the average file download delay of a H-BW and a L-BW leecher respectively. It is easy to see that: $T_H = \frac{S}{R_{downH}}$.

Further, let S_d be the amount of data that a L-BW leecher has downloaded when all H-BW leechers were present in the system. It is easy to see that $S_d = T_H R_{downL}$, and therefore, that the average file download delay of a L-BW leecher can
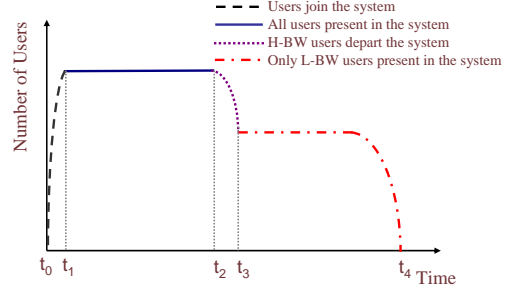
[6]The assumption that $t_0 \approx t_1$ can be justified in a flash crowd scenario. Further, as we shall see in Section VI, the approximation $t_2 \approx t_3$ does not significantly affect the accuracy of the model.

Fig. 2. Evolution of the number of leechers.

be expressed as follows: $T_L = T_H + \frac{S-S_d}{C_{upL}+D_S'}$, where $D_S' = \frac{C_{upS}'}{\alpha N}$ is the average download rate from seeds. Notice that the aggeragate seed capacity C_{upS}' may not be the same as before, since some H-BW leechers might had become seeds instead of leaving the system. In other words, $C_{upS}' = C_{upS} + p_H(1-\alpha)N C_{upH}$, where p_H is the percentage of H-BW leechers that become seeds.

D. *Extension for More Classes of Users*

In this section we show, along the same framework, how to extend our analysis for more classes of users. Now, in addition to L-BW and H-BW leechers, lets consider the third class of leechers: Medium Bandwidth (M-BW) leechers. Further, let α, β, and ζ be the proportion of L-BW, M-BW, and H-BW leechers in the system respectively. Now, it is easy to see that:

$$R_{downH} = n_{HH}^d D_{HH} + n_{HM}^d D_{HM} + n_{HL}^d D_{HL} + D_s, \quad (17)$$
$$R_{downM} = n_{MH}^d D_{MH} + n_{MM}^d D_{MM} + n_{ML}^d D_{ML} + D_s, \quad (18)$$
$$R_{downL} = n_{LH}^d D_{LH} + + n_{LM}^d D_{LM} + n_{LL}^d D_{LL} + D_s. \quad (19)$$

And

$$R_{upH} = n_{HH}^u U_{HH} + n_{HM}^u U_{HM} + n_{HL}^u U_{HL}, \quad (20)$$
$$R_{upM} = n_{MH}^u U_{MH} + n_{MM}^u U_{MM} + n_{ML}^u U_{ML}, \quad (21)$$
$$R_{upL} = n_{LH}^u U_{LH} + n_{LM}^u U_{LM} + n_{LL}^u U_{LL}. \quad (22)$$

Again, to compute the leecher download rates, we first compute the values of upload variables and then relate them to the corresponding download parameters as follows: $D_{LL} = U_{LL}$, $D_{LM} = U_{ML}$, $D_{LH} = U_{HL}$, $D_{ML} = U_{LM}$, $D_{MM} = U_{MM}$, $D_{MH} = U_{HM}$, $D_{HL} = U_{LH}$, $D_{HM} = U_{MH}$, $D_{HH} = U_{HH}$, $n_{LL}^d = n_{LL}^u$, $\alpha n_{LM}^d = \beta n_{ML}^u$, $\alpha n_{LH}^d = \zeta n_{HL}^u$, $\beta n_{ML}^d = \alpha n_{LM}^u$, $n_{MM}^d = n_{MM}^u$, $\beta n_{MH}^d = \zeta n_{HM}^u$, $\zeta n_{HL}^d = \alpha n_{LH}^u$, $\zeta n_{HM}^d = \beta n_{MH}^u$, and $n_{HH}^d = n_{HH}^u$.

To compute the values of upload variables, first, recall that Z is the number of neighbors that a leecher in BitTorrent is uploading to at any time instance. Hence:

$$n_{HH}^u + n_{HM}^u + n_{HL}^u = Z, \quad (23)$$
$$n_{MH}^u + n_{MM}^u + n_{ML}^u = Z, \quad (24)$$
$$n_{LH}^u + n_{LM}^u + n_{LL}^u = Z. \quad (25)$$

Further, under the assumption that a leecher's download link capacity is larger than or equal to his/her upload link capacity, we know that the system's bottlenecks are the upload links

and we can assume that these are fully utilized. This means that $R_{upH} = C_{upH}$, $R_{upM} = C_{upM}$, and $R_{upL} = C_{upL}$. Additionally, as explained earlier, we assume that the upload capacity of a leecher will be fairly shared among concurrent upload connections. Therefore we can write:

$$U_{LL} = U_{LM}, \quad (26)$$
$$U_{LL} = U_{LH}, \quad (27)$$
$$U_{MM} = U_{MH}. \quad (28)$$

Because at any time instance a L-BW leecher is downloading on average from n_{LL}^d L-BW neighbors, we can again compute the *spare* download capacity of this leecher as $C_{downL} - n_{LL}^d D_{LL} - D_S$. And then, the upload rate that a H-BW/M-BW leecher can provide to a L-BW leecher are given by the following two equations:

$$U_{HL} = \min\left(\frac{C_{upH}}{Z}, C_{downL} - n_{LL}^u U_{LL} - D_S\right), (29)$$
$$U_{ML} = \min\left(\frac{C_{upM}}{Z}, C_{downL} - n_{LL}^u U_{LL} - D_S\right) (30)$$

Similarly, we can again compute the *spare* download capacity of a M-BW leecher $C_{spare}^M = C_{downM} - n_{ML}^d D_{ML} - n_{MM}^d D_{MM} - D_S$ and the upload rate that a H-BW leecher can provides to a M-BW leecher is:

$$U_{HM} = \min\left(\frac{C_{upH}}{Z}, C_{spare}^M\right). \quad (31)$$

Denote by $Trinomial(N, p, k)$ the probability mass function of a Trinomial random variable with parameters N, p_1, and p_2 that is, $Trinomial(N, p_1, p_2, k_1, k_2) \equiv \binom{N}{k_1}\binom{N-k_1}{k_2} p_1^{k_1} p_2^{k_2} (1-p_1-p_2)^{(N-k_1-k_2)}$. Recall that H-BW and M-BW leechers will provide uploads to L-BW leechers when they choose a L-BW leecher as the optimistic unchoking neighbor or when they do not sufficient H-BW/M-BW leechers to upload to. Therefore, n_{HL}^u and n_{ML}^u can be computed by Equations (32) and (33) respectively.

$$n_{HL}^u = \sum_{k=0}^{L} \sum_{j=0}^{L-k} n(j,k) P_1(j,k), \quad (32)$$

where:

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

and:

$P_1(j,k) = P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of L}\}$
$= Trinomial(L, \beta, \zeta, j, k).$

$$n_{ML}^u = \sum_{i=0}^{L} \sum_{j=0}^{L-i} n(i,j) P_2(i,j), \quad (33)$$

where:

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

and:

$P_2(i,j) = P\{\text{have } i \text{ L-BW and } j \text{ M-BW neighbors out of L}\}$
$= Trinomial(L, \alpha, \beta, i, j).$

Notice that H-BW leechers will prefer M-BW to L-BW leechers when they do not have sufficient H-BW neighbors to exchange file. Therefore n_{HM}^u is given by the following equation:

$$n_{HM}^u = \sum_{k=0}^{L} \sum_{j=0}^{L-k} n(j,k) P_1(j,k), \quad (34)$$

where:

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

and:

$P_1(j,k) = P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of L}\}$
$= Trinomial(L, \beta, \zeta, j, k).$

Along the same line of analysis in the previous section, we now can compute the TFT rewards for H-BW and M-BW leechers and let $\overline{N}_{unchoke}^{LH}$, $\overline{N}_{unchoke}^{LM}$, and $\overline{N}_{unchoke}^{MH}$ be the average number of times that L-BW, L-BW and M-BW leechers reward H-BW, M-BW, and H-BW leechers respectively. And then, we can compute n_{LH}^u, n_{LM}^u, and n_{MH}^u using Equations (35), (36), and (37) respectively.

$$n_{LH}^u = \sum_{i=0}^{L} \sum_{k=0}^{L-i} n(i,k) P_3(i,k) + \left(\frac{\zeta}{\alpha}\right) n_{HL}^u \frac{\overline{N}_{unchoke}^{LH}}{3}, \quad (35)$$

where:

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

and:

$P_3(i,k) = P\{\text{have } i \text{ L-BW and } k \text{ H-BW neighbors out of L}\}$
$= Trinomial(L, \alpha, \zeta, i, k).$

$$n_{LM}^u = \sum_{i=0}^{L} \sum_{j=0}^{L-i} n(i,j) P_2(i,j) + \left(\frac{\beta}{\alpha}\right) n_{ML}^u \frac{\overline{N}_{unchoke}^{LM}}{3}, \quad (36)$$

where:

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

and:

$P_2(i,j) = P\{\text{have } i \text{ L-BW and } j \text{ M-BW neighbors out of L}\}$
$= Trinomial(L, \alpha, \beta, i, j).$

$$n_{MH}^u = \sum_{j=0}^{L} \sum_{k=0}^{L-j} n(j,k) P_1(j,k) + \left(\frac{\zeta}{\beta}\right) n_{HM}^u \frac{\overline{N}_{unchoke}^{MH}}{3}, \quad (37)$$

where:

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

and:

$$P_1(j,k) = P\{\text{have } j \text{ M-BW and } k \text{ H-BW neighbors out of L}\}$$
$$= Trinomial(L, \beta, \zeta, j, k).$$

V. 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 *all the details* of the BitTorrent protocol and predicts the download rates, and hence the delays, of different classes of users. However, as mentioned before, this model is only applicable in practical cases where the steady state assumption makes sense, *e.g.*, such as in flash crowd scenarios. Further, it does not scale well when there exist many classes of users. (Recall that one needs $2n^2$ equations to characterize a system consisting of n classes of users.)

With the above in mind we now propose a second model–in particular a fluid model. This model can be used to model a more dynamic system where users may arrive and depart at any time instance, *e.g.* such as in non-flash crowd scenarios. Further, it can be easily used to describe a large number of classes of users without becoming complex. It is inspired by the model in [4] (see Section VII for a concrete comparison) and derived along the same lines as our first model, but under the following simplifying approximation/assumption that renders 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. (This 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 VI, the aforementioned approximations do not significantly affect the model's accuracy. However, in steady state scenarios, they render the model slightly less accurate than our first model, which is more detailed (but also more complex). (A detailed comparison between our two models is performed in Section VI.)

As before, we first consider the original BitTorrent system and then proceed with the token-enhanced system.

A. *The Original BitTorrent System*

In the previous section, we have assumed that there are only two classes of users in the system. However, under our above approximation, we can now consider more classes of users while keeping things simple. We say that two users, which can be either leechers or seeds, are in the same class if they have the same link capacity, and we let $\mathbf{G} = \{1, \ldots, K\}$ be the set of user classes in the system.

Denote by $x_j^l(t)$ and $x_j^s(t)$ 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, $\mu_j = \frac{C_{up}^j}{S}$. Further, let $\mu_s(t)$ be the aggregate service rate provided by all seeds in the system at time t. That is, $\mu_s(t) = \sum_{j=1}^K \mu_j x_j^s(t)$. Finally, let $\epsilon_{si}(t)$ and $\epsilon_{ji}(t)$ be the proportion of service rate provided by all seeds and class j leechers respectively, to all class i leechers at time t. And, 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:

$$R_i(t) = \min\left(\sum_{j=1}^K x_j^l(t)\mu_j\epsilon_{ji}(t) + \mu_s(t)\epsilon_{si}(t), x_i^l(t)\frac{C_{down}^i}{S}\right).$$
(38)

Notice that $R_i(t)$ is also the departure rate of class i leechers. Now let $\lambda_i(t)$ be the arrival rate of class i leechers and p_i be the probability that a class i leecher will stay in the system (become a seed) after he/she downloads the file. Then, the population of class i leechers and seeds in the system is described by the following differential equations:

$$x_i^{l'}(t) = \lambda_i(t) - R_i(t), \quad (39)$$
$$x_i^{s'}(t) = R_i(t)p_i - \gamma_i x_i^s(t), \quad (40)$$

where γ_i is the rate at which class i seeds leave the system.

As before, since all leechers in the system are equally likely to be downloading from the seeds, it is easy to see that $\epsilon_{si}(t) = \frac{x_i^l(t)}{\sum_{n=1}^K x_n^l(t)}\mu_s(t)$, $\forall i \in \mathbf{G}$. 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 $\frac{x_i^l(t)}{\sum_{n=1}^K x_n^l(t)}$. Along the same lines of the derivation of Lemma 1, we know that class j leechers cannot upload to class i leechers faster than the spare download capacity of class i leechers $C_{spare}^i = C_{down}^i - \epsilon_{ii}(t)\mu_i - \frac{\epsilon_{si}(t)\mu_s(t)}{x_i^l(t)}$, nor faster than the fair share that class i leechers can receive, which is $\frac{\mu_j}{Z}$. (Recall that Z is the configured number of user upload connections.)

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

***Lemma* 7:**

$$\epsilon_{ji}(t) = \begin{cases} \frac{x_i^l(t)}{\sum_{n=1}^K x_n^l(t)} \min\left(\frac{1}{Z}, \frac{C_{spare}^i}{\mu_j}\right) & \text{if } i \neq j, \\ 1 - \sum_{i=1, i\neq j}^K \epsilon_{ji}(t) & \text{otherwise.} \end{cases}$$
(41)

For a system with K classes of users we have $2K$ variables ($\{x_i^l(t)\}$, $\{x_i^s(t)\}, i \in \mathbf{G}$) and $2K$ differential equations (two for each class, like Equations (39) and (40)), that dictate the evolution of these $2K$ variables. Therefore, we can solve this system of equations to study how the user population ($\{x_i^l(t)\}$, $\{x_i^s(t)\}, i \in \mathbf{G}$) 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.

B. The Token-enhanced BitTorrent System

It is easy to see that Equations (38)...(40) still hold in the token-enhanced system. What changes in the token-enhanced system 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 capacity, with class 1 be the class with the lowest capacity. Along the same lines of the analysis in Section IV-B, 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 $\epsilon_{1i}(t) = \frac{x_i^l(t)}{\sum_{n=1}^{K} x_n^l(t)}, \forall i \in \mathbf{G}$. 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 $\epsilon_{12}(t)\mu_1 K_{up}$, and the token spending rate of the class 1 leecher to the class 2 leecher is $\epsilon_{21}(t)\mu_2 K_{down}$. Because in the long run the token earning rate of all class 1 leechers from class 2 leechers ($x_1^l(t)\epsilon_{12}(t)\mu_1 K_{up}$) equals the token spending rate of all class 1 leechers to class 2 leechers ($x_2^l(t)\epsilon_{21}(t)\mu_2 K_{down}$), we can write $\epsilon_{21}(t) = \frac{x_1^l(t)\epsilon_{12}(t)\mu_1 K_{up}}{x_2^l(t)\mu_2 K_{down}}$. 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 $\frac{x_1^l(t)}{\sum_{n=1}^{K} x_n^l(t)}$. [7] Therefore:

$$\epsilon_{21}(t) = \min \left(\frac{x_1^l(t)\epsilon_{12}(t)\mu_1 K_{up}}{x_2^l(t)\mu_2 K_{down}}, \frac{x_1^l(t)}{\sum_{n=1}^{K} x_n^l(t)} \right). \quad (42)$$

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). Hence:

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

Continuing this way, it is easy to see that a general formula for $\epsilon_{ji}(t)$ is the following:

Lemma 8:

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

$$(43)$$

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.

VI. EXPERIMENTS

We use an event driven BitTorrent simulator developed by [27] to validate our analysis. The detailed simulator description

---

[7] It is interesting to point out that this amount is independent of Z because a class 2 leecher on average uploads to $Z \frac{x_1^l(t)}{\sum_{n=1}^{K} x_n^l(t)}$ class 1 leechers with rate $\frac{\mu_2}{Z}$.

---

can be found in [17]. In addition, we implement the proposed token based scheme to study its impact on the system performance.

A. Steady State Performance Prediction and Flash Crowd Scenarios

To validate the model of Section IV, we first simulate a flash crowd scenario where 200 leechers join the system within 20 seconds. Leechers will leave the system as soon as they finish their download. We simulate the system until all leechers depart. 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 256KB, and (iii) the maximum number of concurrent upload transfers (Z) is 5 (the default value).

We present simulation results for two scenarios, which, as we shall see, yield qualitatively different results when the token based scheme is used. In both scenarios the percentage of L-BW leechers is $\alpha = 0.8$, $C_{downH} = 600$Kbps, and $C_{upH} = 300$Kbps. For Scenario 1 we have that $C_{downL} = 300$Kbps, $C_{upL} = 100$Kbps, and for Scenario 2 we have $C_{downL} = 150$Kbps, $C_{upL} = 50$Kbps.

1) Simulation Results for the Original BitTorrent System: In Figures 3 and 4 we present both theoretical and simulation results for Scenarios 1 and 2 respectively. Figures 3(i) and 4(i) show how n_{HL}^u, the average number of L-BW leechers that are downloading from a H-BW leecher, behaves as the number of neighbors (L) increases. (These plots will help in gaining intuition when we explain how the leecher download rates and delays change as a function of L.) First, from these plots we can see that Equation (9) can correctly predict n_{HL}^u. Further, we can observe that n_{HL}^u 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 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 leechers.

The download rates for both H-BW and L-BW leechers with respect to L are shown in Figures 3(ii) and 4(ii). Note that these results correspond to the period $(t_1, t_2]$ in Figure 2, where both classes of leechers are present in the system. (Recall that in the interval $(t_3, t_4]$, where there are no H-BW leechers, the download rate of L-BW leechers is just equal to their upload capacity plus the rate that they can download from seeds.) Again, we can observe from the plots that our mathematical model is quite accurate. Notice that the download rate of H-BW leechers increases and the download rate of L-BW leechers decreases as L increases. This can be explained in a similar manner like we did with n_{HL}^u. As L increases H-BW leechers provide uploads to fewer L-BW leechers and to more H-BW leechers.

Finally, the results for the average file download delay for the two scenarios are shown in Figures 3(iii) and 4(iii). We can observe that our model is quite accurate in predicting the average file download delay for H-BW leechers, L-BW leechers, and for the whole system. (The small discrepancies in the delay prediction are because we are ignoring the short transient periods, as explained earlier (in Section IV-C).)
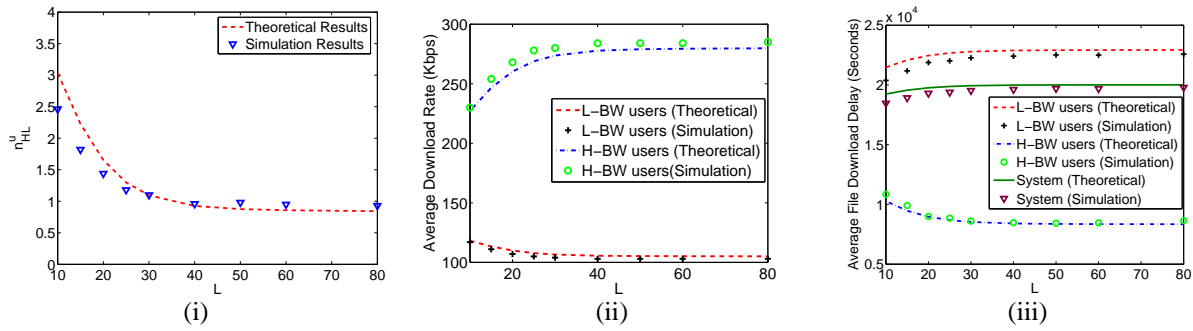
Fig. 3. Scenario 1: (i) Average number of L-BW leechers that a H-BW leecher is uploading to, (ii) Average download rate for H-BW and L-BW leechers, (iii) Average file download delay for H-BW leechers, L-BW leechers, and for the system. (Original BitTorrent system.)
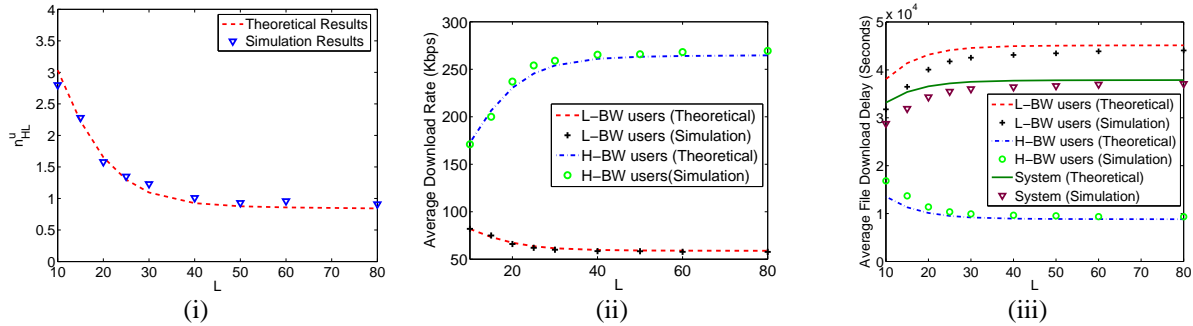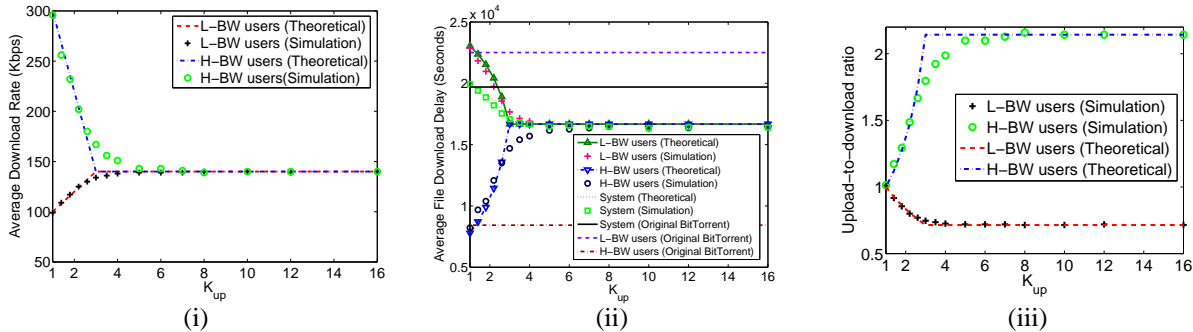

Fig. 4. Scenario 2: (i) Average number of L-BW leechers that a H-BW leecher is uploading to, (ii) Average download rate for H-BW and L-BW leechers, (iii) Average file download delay for H-BW leechers, L-BW leechers, and for the system. (Original BitTorrent system.)
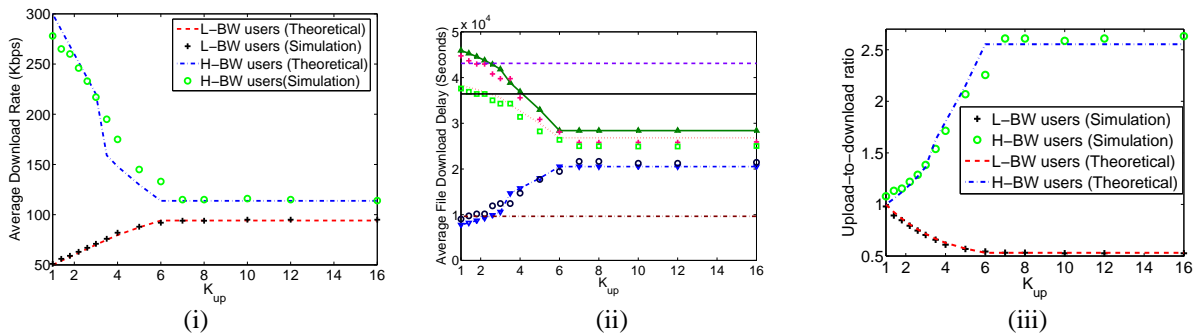

Fig. 5. Scenario 1: (i) Average download rate for H-BW and L-BW leechers, (ii) Average file download delay for H-BW leechers, L-BW leechers, and for the system, (iii) Upload-to-download ratio for H-BW and L-BW leechers. (Token-enhanced system.)

2) Simulation Results for the Token-enhanced BitTorrent System: We now let $K_{down} = 1$ and study how the token-enhanced system behaves for different values of K_{up} in the two scenarios we described earlier. We fix $L = 40$, which is a typical value in BitTorrent [3]. Figures 5 and 6 show theoretical and simulation results for Scenarios 1 and 2 respectively. (We will explain shortly what we mean by "upload-to-download ratio" in Figures 5(iii) and 6(iii).)

The download rates for both classes of leechers are shown in Figures 5(i) and 6(i). From the plots 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. (Hence

their download rate increases.) 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. Further, it is interesting to point out that in the first scenario the two classes of leechers have the same (constant) download rate for large K_{up}, whereas in the second scenario the download rates of the two classes are never equal. This is because in the first scenario (for large K_{up}) both classes of leechers are downloading from a similar number of H-BW leechers, and since $C_{downL} = C_{upH}$ and $C_{downH} > C_{upH}$, both classes of leechers can fully utilize the H-BW leechers' upload capacity. In contrast, in the second scenario, while both classes of leechers are downloading from a similar number of H-BW leechers (for large K_{up}), since $C_{downH} > C_{upH}$ but $C_{downL} < C_{upH}$, only H-BW leechers can fully utilize the

Fig. 6. Scenario 2: (i) Average download rate for H-BW and L-BW leechers, (ii) Average file download delay for H-BW leechers, L-BW leechers, and for the system, (iii) Upload-to-download ratio for H-BW and L-BW leechers. (Token-enhanced system.)

upload capacity of other H-BW leechers from whom they are downloading.

Figures 5(ii) and 6(ii) show results for the average file download delay for H-BW leechers, L-BW leechers, and for the whole system. For comparison, the plots also show 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 $K_{up} = 1 = K_{down}$, the performance of the token-enhanced system is almost identical to that of the original BitTorrent system. However, as K_{up} increases the overall system performance can be improved compared to the original BitTorrent system. This is because in the token-enhanced system L-BW leechers are downloading from more H-BW leechers if K_{up} is large, since as we have mentioned earlier, L-BW leechers can gain tokens fast. However, as mentioned earlier, we are sacrificing the perceived performance of H-BW leechers. This motivates us to quantify next how much "unfair" the token based scheme becomes to H-BW leechers as K_{up} increases.

3) Impact of the Proposed Token Based Scheme on 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. [8] Figures 5(iii) and 6(iii) show how the upload-to-download ratio behaves as we vary K_{up}, for each class of users.

From these plots we observe that the upload-to-download ratio is almost the same for both classes of leechers when $K_{up} = 1 = K_{down}$. 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.)

Looking at Figures 5(ii) (6(ii)) and 5(iii) (6(iii)) we can conclude that we can tradeoff between overall system performance and fairness. 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 $K_{up} = 1 = K_{down}$, 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}.

[8]This metric has been also used to quantify fairness in other studies as well, *e.g.* [17], [18].

B. 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, in order to demonstrate how accurate the fluid model of Section V is.

We consider three classes of users: H-BW, M-BW, and L-BW users. We consider again two different scenarios in order to validate the model under different settings. In the first scenario we have $C_{downH} = 700$Kbps, $C_{upH} = 700$Kbps, $C_{downM} = 700$Kbps, $C_{upM} = 300$Kbps, $C_{downL} = 700$Kbps, and $C_{upL} = 100$Kbps. Further, all three classes of leechers arrive to the system at the same rate, which remains constant throughout the experiment. In the second scenario, we have $C_{downH} = 1000$Kbps, $C_{upH} = 600$Kbps, $C_{downM} = 1000$Kbps, $C_{upM} = 250$Kbps, $C_{downL} = 1000$Kbps, and $C_{upL} = 100$Kbps. Further, now, the arrival rates of the three classes of leechers are different and change during the simulation. This is in order to simulate an even more dynamic system. The detailed leecher arrival rates for the two scenarios are shown in Figures 7(i) and 8(i). Finally, we fix $L = 60$ (which is also a typical value in BitTorrent), the file size is 100MB, 15% of leechers will stay in the system for 3000 seconds after they finish their download (i.e. $p_L = p_M = p_H = 0.15$ and $\gamma_L = \gamma_M = \gamma_H = \frac{1}{3000}$), and all other simulation parameters are the same as before.

1) Simulation Results for the Original BitTorrent System: Figures 7(ii) and 8(ii) show how the number of users (seeds and leechers) in the system evolves over time, for Scenarios 1 and 2 respectively. From the plots we can observe that the proposed fluid model can capture in general the simulation results quite accurately. 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 that 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. Clearly,
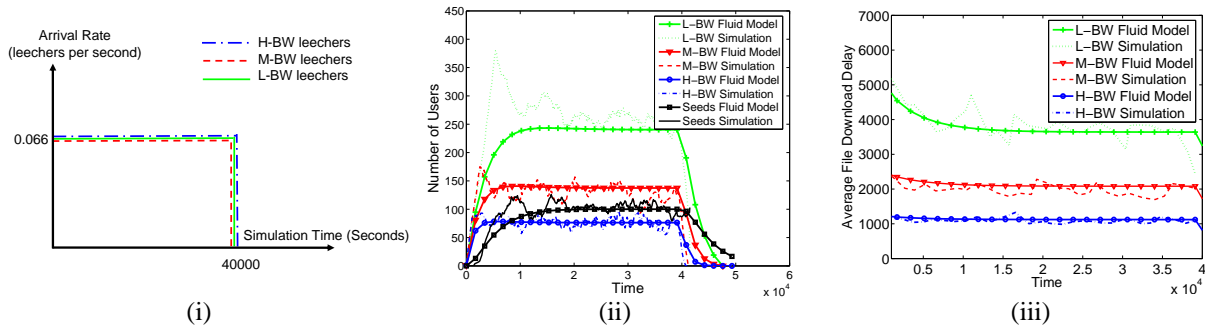
Fig. 7. Scenario 1: (i) Leecher arrival rate, (ii) Number of users in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers. (Original BitTorrent system.)
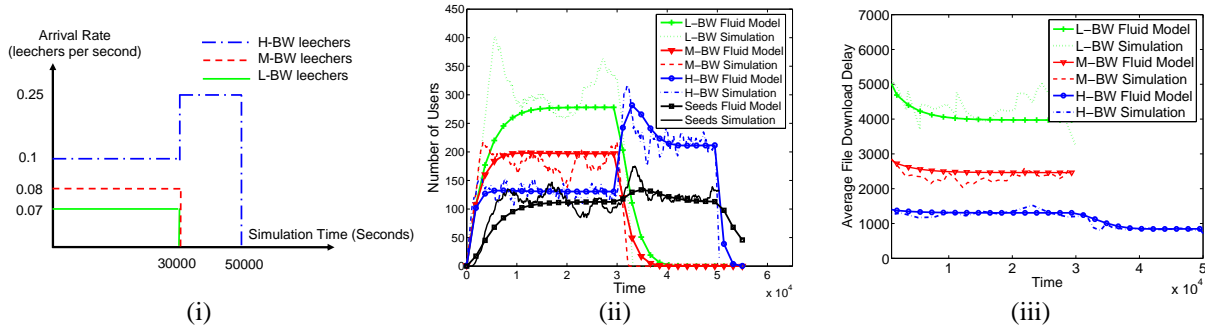

Fig. 8. Scenario 2: (i) Leecher arrival rate, (ii) Number of users in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers. (Original BitTorrent system.)

after leechers stop arriving to the system the number of leechers starts decreasing. Due to space limitations we do not present the evolution of seeds. One can make similar observations from the plots. Before proceeding, note 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 Scenario 2, as shown in Figure 8(i). From Figure 8(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 are because the model does not consider the fact that leechers initially require a large amount of time to finish their download, and hence to depart the system. In particular, Equation (38) does not consider the fact that initially the leecher departure rate may be zero, but instead, it always assumes that this is strictly positive. This, in turn, means that the leecher departure rate is initially overestimated in the fluid model, and as a consequence, the rate by which the number of leechers in the system increases is lower than the one in the simulation.

Finally, Figures 7(iii) and 8(iii) show 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. First, notice that the file download delays initially decrease as time increases because the number of seeds in the system initially increases (and thus leechers can receive more downloads from seeds). After the number of seeds in the system stabilizes the file download delays also stabilize as expected. Further, the discrepancies between theoretical and simulation results in these plots are

due to the approximations that took place in the fluid model. In particular, they are due to the fact that the model does not consider the optimistic unchoking reward scenario. This implies two things: (i) the download delays of higher bandwidth leechers are being overestimated, since we are ignoring the download rate rewards that these leechers receive when they optimistically unchoke lower bandwidth leechers, and (ii) the download delays of lower bandwidth leechers are being underestimated, since lower bandwidth leechers do no longer spend 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, these discrepancies are relatively small, as we can deduce by looking at Figures 7(iii) and 8(iii). We believe that this is a good tradeoff between the achieved accuracy and the simplicity/generality of the model. (We will shortly discuss this issue in more detail.)

2) Simulation Results for the Token-enhanced BitTorrent System: We fix $K_{down} = 1$ and vary K_{up} to study its effect on the system dynamics. Figure 9 presents analytical and simulation results for the peer population in the token-enhanced system for Scenario 1. In the figure we show results for $K_{up} = 1$, $K_{up} = 2$, and $K_{up} = 15$.

Again we observe that our model is quite accurate. Further, we can see that the token-enhanced system (Figure 9(i)) can resemble the original BitTorrent system (Figure 7(ii)) when $K_{up} = 1 = K_{down}$, in agreement with our earlier arguments. In addition we can observe that as we increase K_{up}, the number of lower bandwidth leechers in the system decreases and of higher bandwidth leechers increases. The explanation
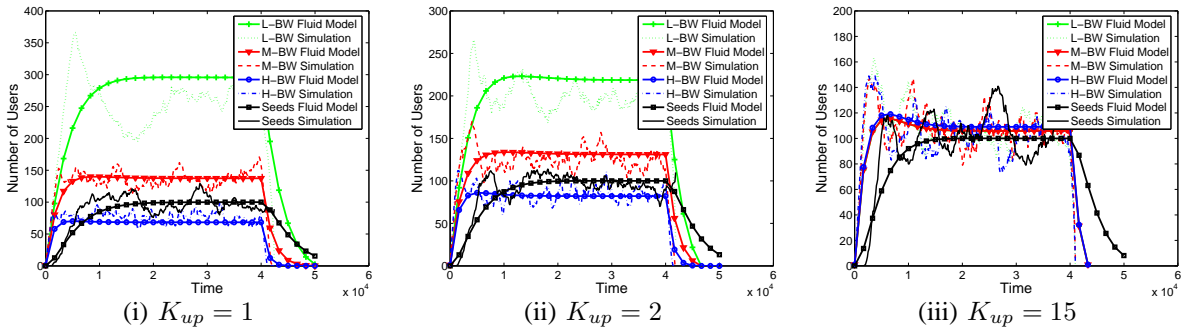
Fig. 9. Scenario 1: Number of users in the system. (Token-enhanced system.)

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. Notice that in this scenario the number of leechers present in the system from each class, is the same when $K_{up} = 15$. This is because, as explained earlier, for large K_{up} leechers can download, and hence depart the system, at the same rate (as long as they are not constrained by their download capacity). This is reminiscent of the situation we saw earlier in Figure 5(i).

C. Comparison Between the Two Models

As we have seen, our first model (presented in Section IV) is tailored for steady state performance analysis, accounts for all 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 VI. We then solve our fluid model equations in steady state (i.e. in the interval $[t_1, t_2)$ in Figure 2), and compute the corresponding leecher download rates.

The results are depicted in Figure 10. As we observe, our first model is more accurate as expected, since it captures more system details than the second model. In particular, it captures the effect of connecting to a variable number of neighbors (L), and considers the optimistic unchoking reward scenario. Our second model is less accurate when L is small, because it does not consider the fact that leechers may not 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, as we explained earlier. It is interesting to point out that this difference for H-BW leechers never exceeds $\frac{100}{Z}\%$. 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 $\frac{100}{Z}\%$ of its upload link capacity (see Equation (8)). 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 $\frac{100}{Z}\%$. (Recall that $Z = 5$ for the scenarios we considered here, and notice that as Z increases the discrepancy decreases here.) Now, consider that total amount of reward that H-BW leechers do not receive equals the total amount of extra download that L-BW leechers receive. Therfore, our second model overestimate the download rate of L-BW leechers by $\frac{(1-\alpha)C_{upH}}{Z\alpha}$. As a result, our model may overestimate the rate of L-BW leechers by at most $\frac{(1-\alpha)C_{upH}100}{C_{upL}Z\alpha}\%$.

However, 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.

VII. 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: [4] and [8]. The reason of choosing these two results is that [4] presents the first mathematical model for BitTorrent systems and [8] is one of the most representative works that considers network heterogeneity. We refer to the model proposed in [4] as the DR's model and to the model proposed in [8] as the FPK's model. The comparison is summarized in Table I. (The table also gives a summary of the differences between our two models, which we discussed earlier.)

From Table I we see that among all models, our first model is the most inclusive one. It models all details of a BitTorrent-like system, except from the system time dynamics. In particular, 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), network 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 DR's and FPK's models (as well as our second model), consider the system's time dynamics but
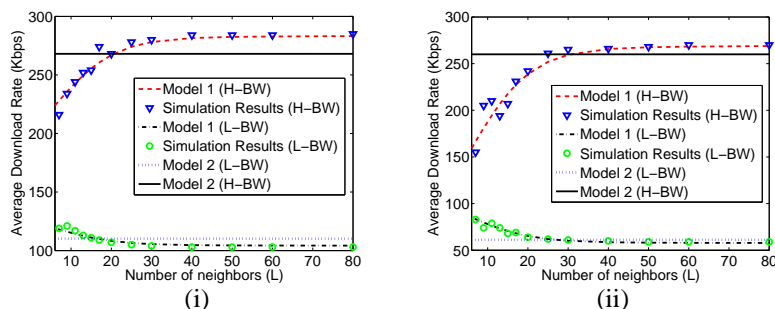
Fig. 10. Average download rate of H-BW and L-BW leechers: (i) Scenario 1, and (ii) Scenario 2. (Comparison between the two models, original BitTorrent.)

System details captured by the Model	Model 1	Model 2	FPK's model	DR's model
number of concurrent uploads (Z)	Yes	Yes	No	No
number of neighbors (L)	Yes	No	No	No
number of user classes (heterogeneity)	many	many	2	1
performance 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

TABLE I
MODEL COMPARISON.

under some approximations/simplifications. The DR's model only considers system time dynamics in a homogeneous BitTorrent system, where all users have the same capacities. The FPK's model incorporates network heterogeneity (two classes of users) into the DR's model. It also attempts to model the effect of the optimistic unchoking and of the TFT scheme. However, it does not correctly capture how these affect system performance in realistic scenarios. (That is why in the corresponding entry of Table I we write "Partial".) 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. 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 and it is the most general one. In particular, it accounts for all the system details, except from the variability in the number of neighbors that a user might be connected to (L), and the optimistic unchoking reward scenario. Finally, to our best knowledge, this is also the first model that considers an arbitrary number of user classes in heterogeneous environments.

VIII. 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 can be used to predict the average file download delay among users with different capacities in steady state scenarios. The second model is a fluid model, which can be used to study both system performance and time dynamics. We have seen that

there is a tradeoff in the two models between accuracy and simplicity/generality.

Further, we have proposed a flexible token based TFT scheme that can be used to tradeoff between fairness and system performance. We have extended our mathematical models in order to predict the system performance under the proposed token based scheme and for tuning the scheme's parameters. Our results have been verified using extensive simulations.

There are some interesting directions for future work. First, we plan to use our models to thoroughly investigate if there are combinations for the values of the system's parameters that achieve optimal download rates/delays. Second, we plan to further generalize our fluid model to incorporate the optimistic unchoking reward scenario, the fact that users may leave the system before finishing their downloads, as well as relaxing the assumption that users always have sufficient neighbors of the same class to connect to.

APPENDIX

A. *More Results For The Steady State Model*

1) Two Classes of Users: In this section, we provide our extensive simulation results for the model 1. In all simulations, the link capacity of H-BW leechers are kept the same as described before. Table II summarizes corresponding link capacity of L-BW leechers. The results are shown in Figures 11...16.

2) Three Classes of Users: Now we present simulation results to validate the extended model in Section IV-D. We assume three classes of users: L-BW, M-BW, and H-BW users. Again we simulate two different scenarios. In both scenario, the upload/download link capacities of L-BW, M-BW, and H-BW users are 50Kbps/900Kbps, 200Kbps/900Kbps, and 800Kbps/900Kbps respectively. In the first scenario we have
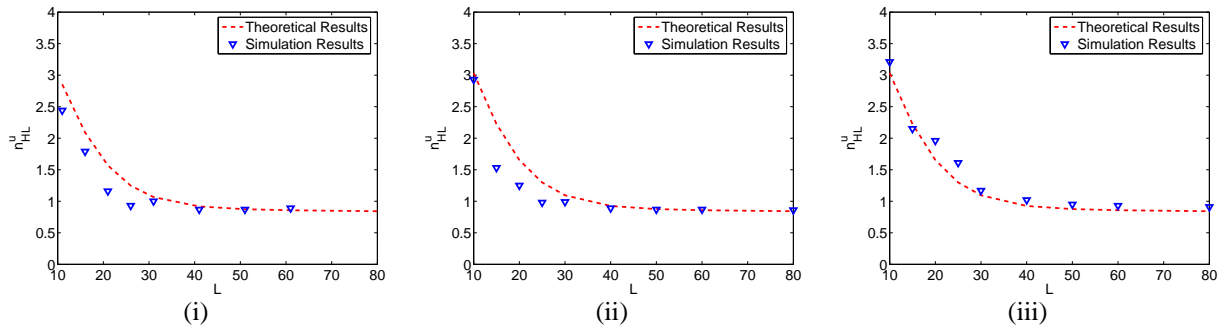
Fig. 11.   Average number of L-BW leechers that a H-BW leecher is uploading to: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.
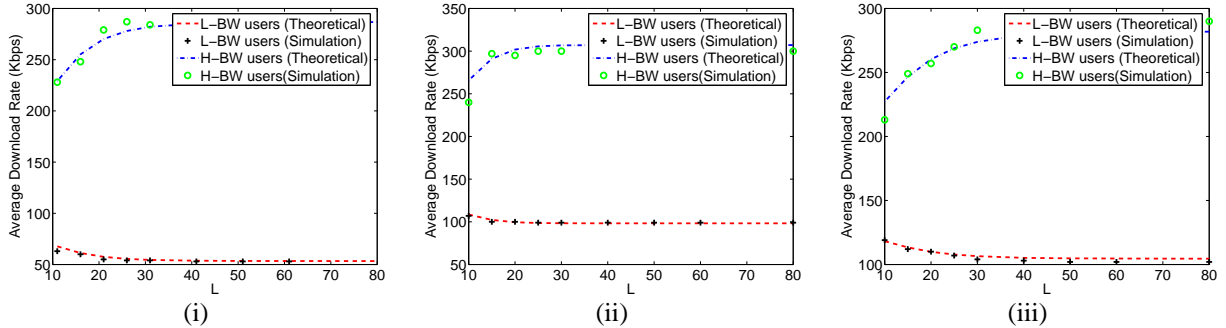

Fig. 12.   Average download rate for H-BW and L-BW leechers: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.
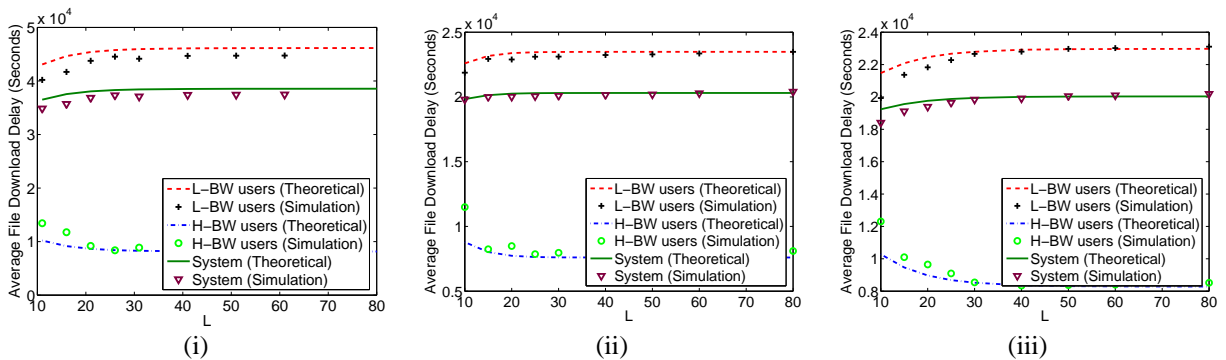

Fig. 13.   Average file download delay for H-BW leechers, L-BW leechers, and for the system: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.

Scenario	Upload (Kbps)	Download (Kbps)
I	50	80
II	100	120
III	100	150

TABLE II
UPLOAD AND DOWNLOAD BANDWIDTH USED IN THE SIMULATION.

$\alpha = \beta = 0.4, \zeta = 0.2$ and $\alpha 0.4, \beta = \zeta = 0.3$ in the second scenario. All other simulation parameters are the same as before. Figure 17 shows that our model can accurately predict the simulation results.

B. *More Results for The Fluid Model*

We now present more results to validate our fluid model. We again consider three classes of leechers: H-BW, M-BW, and L-BW leechers, and present two more different scenarios. In the first scenario we have $C_{downH} = 700$Kbps, $C_{upH} = 700$Kbps, $C_{downM} = 700$Kbps, $C_{upM} = 300$Kbps, $C_{downL} = 700$Kbps, and $C_{upL} = 100$Kbps. In the second scenario, we have $C_{downH} = 900$Kbps, $C_{upH} = 800$Kbps, $C_{downM} = 900$Kbps, $C_{upM} = 200$Kbps, $C_{downL} = 900$Kbps, and $C_{upL} = 50$Kbps. Finally, leechers will leave the system after they finish their download (i.e. $p_L = p_M = p_H = 0$), and all other simulation parameters are the same as before. Figures 18 and 19 show the results for Scenarios 1 and 2 respectively.

REFERENCES

[1] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," in *Proc. of the Fifth Symposium on Operating System Design and Implementation (OSDI)*, 2002.

[2] A. Parker, "The true picture of peer-to-peer filesharing," http://www.cachelogic.com/ (accessed October 2007).

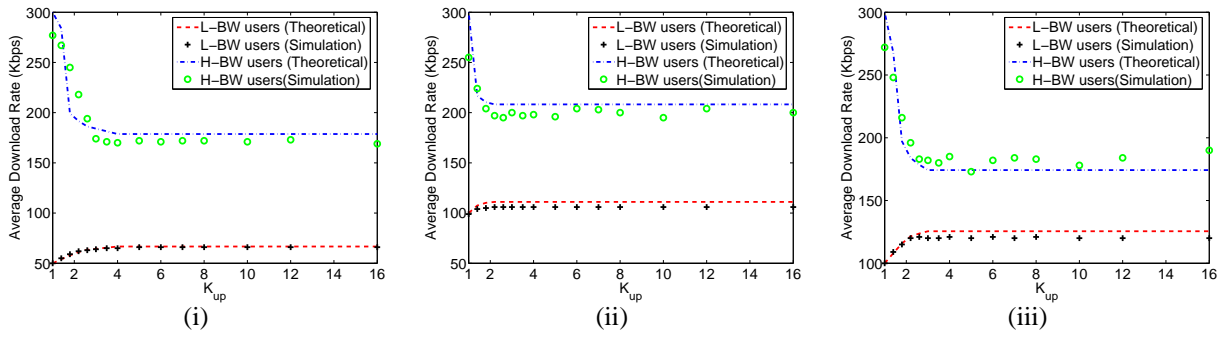[3] "Bittorrent," http://www.bittorrent.com/protocol.html (accessed October 2007).

Fig. 14. Average download rate for H-BW and L-BW leechers: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.
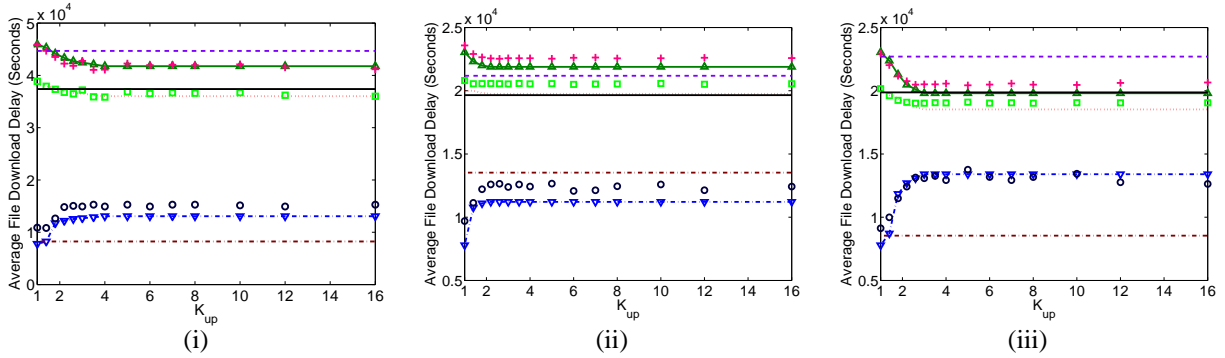

Fig. 15. Average file download delay for H-BW leechers, L-BW leechers, and for the system: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.
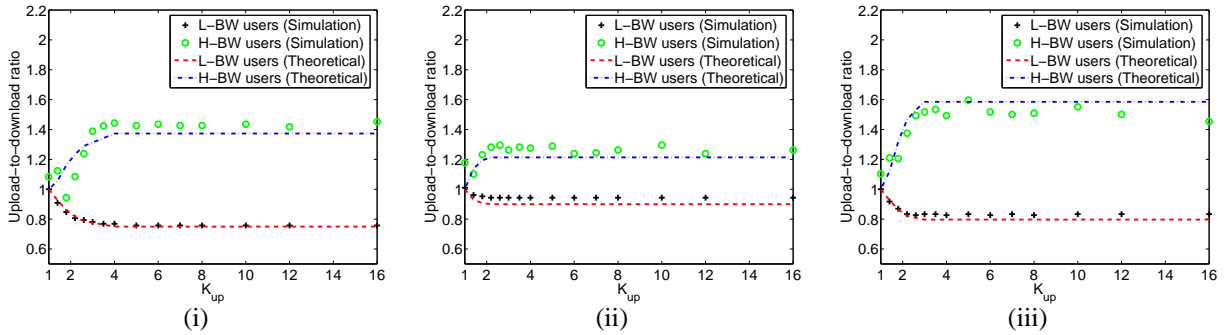

Fig. 16. Upload-to-download ratio for H-BW and L-BW leechers: (i) Scenario 1, (ii) Scenario 2, and (iii) Scenario 3.

[4] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. of ACM SIGCOMM*, 2004.

[5] F. Piccolo and G. Neglia, "The effect of heterogeneous link capacities in bittorrent-like file sharing systems," in *Proc. of 1st International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P '04)*, 2004.

[6] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the role of helpers in peer-to-peer file download systems: Design, analysis, and simulation," in *Proc. of 6th International Workshop on Peer-to-Peer Systems (IPTPS '07)*, 2007.

[7] B. Fan, D.-M. Chiu, and J. Lui, "Stochastic differential equation approach to model bittorrent-like file sharing systems," in *Proc. of 14th IEEE International Workshop on Quality of Service*, 2006.

[8] F. Clevenot, P. Nain, and K. Ross, "Multiclass p2p networks: Static resource allocation for service differentiation and bandwidth diversity," in *Proc. of 24th IFIP WG 7.3 International Symposium on Computer Performance, Modeling, Measurements and Evaluation*, 2005.

[9] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of bittorrent-like peer-to-peer systems," in *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 1, January 2007.

[10] Y. Tian, D. Wu, and K. Ng, "Modeling, analysis and improvement for bittorrent-like file sharing networks," in *Proc. of IEEE INFOCOM*, 2006.

[11] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," in *Proc. of Internet Measurement Conference*, 2005.

[12] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting bittorrent for fun (but not profit)," in *Proc. of 5th International Workshop on Peer-to-Peer Systems (IPTPS '06)*, 2006.

[13] D. Hales and S. Patarin, "How to cheat bittorrent and why nobody does," University of Bologna, Italy, Tech. Rep. UBLCS-2005-12, 2005.

[14] "Bigchampagne," http://www.bigchampagne.com/ (accessed October 2007).

[15] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS '05)*, 2005.

[16] M. Izal, G. Keller, E. Biersack, P. Felber, A. Hamra, and L. Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," in *Proc. of Passive and Active Measurements*, 2004.

[17] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving bittorrent performance," in *Proc. of IEEE INFOCOM*, 2006.

[18] R. Thommes and M. Coates, "Bittorrent fairness: analysis and improvements," in *Proc. of Workshop Internet, Telecom. and Signal Proc.*, December 2005.
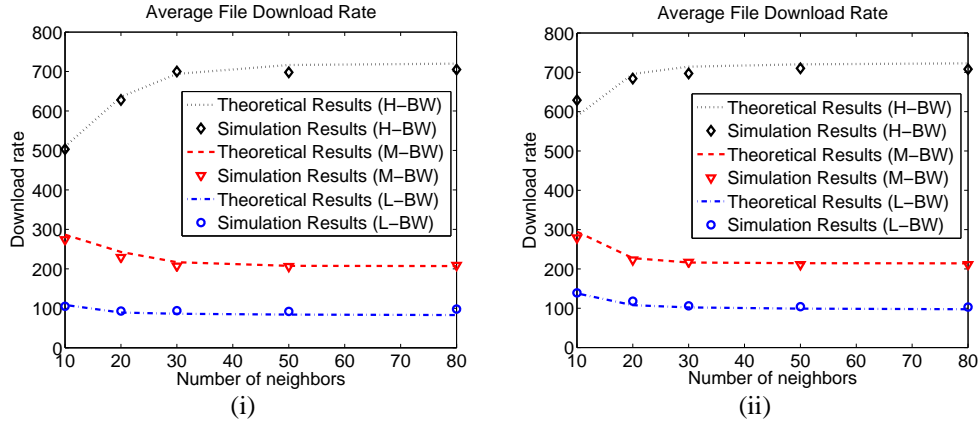
Fig. 17. Download rates for H-BW, M-BW, and L-BW leechers: (i) Scenario 1 and (ii) Scenario 2.
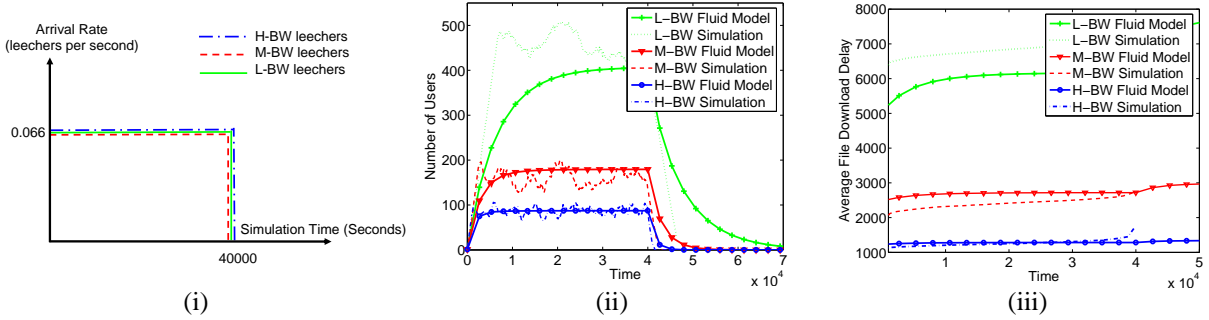

Fig. 18. Scenario 1: (i) Leecher arrival rate, (ii) Number of leechers in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers. (Original BitTorrent system.)
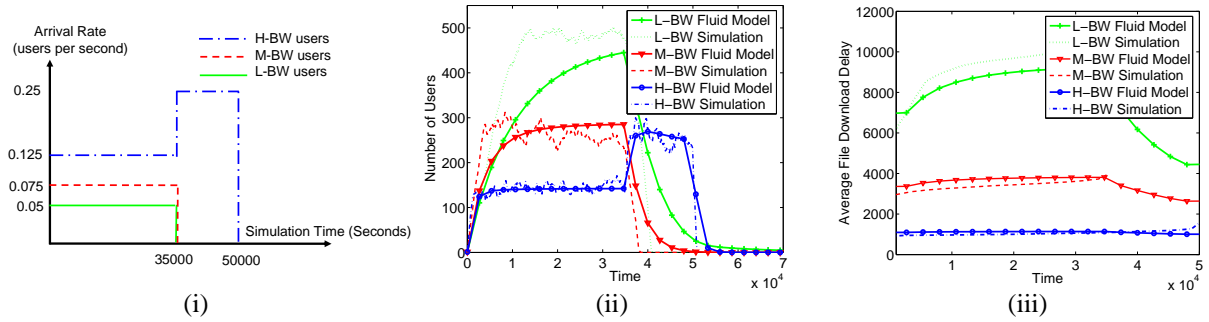

Fig. 19. Scenario 2: (i) Leecher arrival rate, (ii) Number of leechers in the system, and (iii) Average file download delay for H-BW, M-BW, and L-BW leechers. (Original BitTorrent system.)
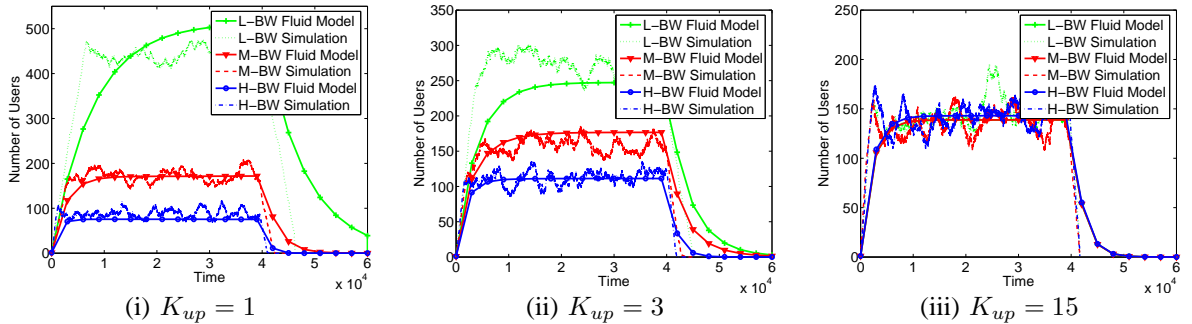

(i) $K_{up} = 1$ (ii) $K_{up} = 3$ (iii) $K_{up} = 15$

Fig. 20. Scenario 1: Number of leechers in the system. (Token-enhanced system.)

[19] F. Bin, D.-M. Chiu, and J. C. Lui, "The delicate tradeoffs in bittorrent-like file sharing protocol design," in *Proc. of International Conference on Network Protocols (ICNP),*, November 2006.

[20] W.-C. Liao, F. Papadopoulos, and K. Psounis, "An efficient algorithm for

resource sharing in peer-to-peer networks," in *Proc. of IFIP Networking*, 2006.

[21] ——, "A peer-to-peer cooperation enhancement scheme and its performance analysis," in *Journal of Communications*, vol. 1, no. 7, November 2006.

[22] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[23] X. Yang and G. D. Veciana, "Service capacity of peer to peer networks," in *Proc. of IEEE INFOCOM*, 2004.

[24] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno, "Analysis of resource transfers in peer-to-peer file sharing applications using fluid models," in *Performance Evaluation*, vol. 63. Elsevier, 2006, pp. 149–174.

[25] S. Saroiu, K. Gummadi, R. Dunn, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[26] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do Incentives Build Robustness in BitTorrent? (Awarded Best Student Paper)," in *NSDI*, 2007.

[27] A. Bharambe, C. Herley, and V. Padmanabhan, "Microsoft research simulator for the bittorrent protocol," http://www.research.microsoft.com/projects/btsim (accessed October 2007).