

Modeling and Analysis of Bandwidth-Inhomogeneous Swarms in BitTorrent

M. Meulpolder, J.A. Pouwelse, D.H.J. Epema, H.J. Sips
Parallel and Distributed Systems Group
Department of Computer Science
Delft University of Technology, The Netherlands
m.meulpolder@tudelft.nl

Abstract—A number of analytical models exists that capture various properties of the BitTorrent protocol. However, until now virtually all of these models have been based on the assumption that the peers in the system have *homogeneous* bandwidths. As this is highly unrealistic in real swarms, these models have very limited applicability. Most of all, these models implicitly ignore BitTorrent's most important property: peer selection based on the *highest rate* of reciprocity. As a result, these models are not suitable for understanding or predicting the properties of real BitTorrent networks. Furthermore, they are hardly of use in the design of realistic BitTorrent simulators and new P2P protocols. In this paper, we extend existing work by presenting a model of a swarm in BitTorrent where peers have arbitrary upload and download bandwidths. In our model we group peers with (roughly) the same bandwidth in *classes*, and then analyze the allocation of upload slots from peers in one class to peers in another class. We show that our model accurately predicts the bandwidth clustering phenomenon observed experimentally in other work, and we analyze the resulting data distribution in swarms. We validate our model with experiments using real BitTorrent clients. Our model captures the effects of BitTorrent's well-known 'tit-for-tat' mechanism in bandwidth-inhomogeneous swarms and provides an accurate mathematical description of the resulting dynamics.

I. INTRODUCTION

Over the last decade, P2P file-sharing systems have become very popular and mathematical models of such systems have been proposed in a variety of forms. Much research has been focused on modeling the properties of P2P file sharing systems in general, such as [1], [2], [3], [4]. Other research has modeled specific applications, strategies, or protocols, such as the BitTorrent protocol in [5], [6], [7]. However, in virtually all of this work, mathematical descriptions have only been derived for systems with *homogeneous* bandwidths of peers. While this is a convenient assumption in theoretical analysis, it is completely unrealistic in practice. In this paper, we significantly extend the existing line of work by providing a detailed mathematical model for BitTorrent swarms with peers of *arbitrary* bandwidths.

The BitTorrent protocol [8] is one of today's most popular P2P protocols used by millions of users worldwide. The most essential properties of BitTorrent are: (i) peers form swarms where *pieces of the same file* are exchanged; (ii) pieces are exchanged in a *tit-for-tat* manner, where a peer uploads pieces to those peers that return pieces *with the highest rate*; (iii) peers employ *rarest-first* piece selection to ensure efficient spreading of the file contents in the

swarm. The models that target BitTorrent in particular, such as the well known model of Qiu *et al.* [7], assume that all peers in a swarm have the same bandwidth. By making this assumption, peer selection becomes arbitrary and hence one of BitTorrent's most essential properties is rendered irrelevant. In the Internet, homogeneity can obviously not be assumed. While some general P2P models do assess arbitrary bandwidth configurations [2], [4], they are not compatible with BitTorrent's swarm based piece-exchange mechanism. Legout *et al.* [9] have shown by experiment that in BitTorrent swarms, clusters emerge of peers with more or less the same bandwidth. However, while they observe this experimentally they do not provide a mathematical model that predicts the results a-priori. Furthermore, strategies exist that specifically exploit the rate-based peer selection of BitTorrent [10], yet can only be assessed experimentally for lack of applicable models.

Our aim is to provide a model that enables the research community to: (i) understand, assess, and predict the properties of BitTorrent systems in practice; (ii) create a layer of abstraction in simulations of BitTorrent systems; (iii) design new P2P protocols that improve the download rates and 'fairness' resulting from BitTorrent's tit-for-tat mechanism. In this paper, we provide the following contributions:

- 1) We extend the line of work in [7], [6] and provide a fluid model that incorporates the dynamics of peers with different bandwidths based on a detailed analysis of BitTorrent's *unchoke policy*.
- 2) We analyze the upload slot allocation in bandwidth-inhomogeneous swarms and show that our model theoretically predicts the bandwidth clustering in BitTorrent that was previously only observed experimentally [9].
- 3) We show that the bandwidth clustering based on upload slot allocation is not necessarily representative for the actual data distribution in the swarm. While in our scenarios fast peers allocate most of their upload slots to other fast peers, they are still responsible for most of the data downloaded by slower peers.

In our analysis, we group peers with (roughly) the same bandwidth in *classes*. Then, we model the allocation of the upload slots of the peers over the various classes. For clarity, we start with analyzing a swarm with two classes (i.e., fast peers and slow peers). We then generalize our analysis to swarms with N classes, hence providing a suitable approach to model any arbitrary swarm. We

validate our model with experiments with real BitTorrent clients. We show that for swarms with various proportions of fast, respectively, slow peers, our theoretical predictions very accurately match the real results. With our model, we hope to open a range of new possibilities in the analysis, simulation, and design of BitTorrent systems in practice.

The remainder of this paper is organized as follows. In Section II, we provide an overview of the BitTorrent protocol and in particular its unchoke policy. In Section III, we present our model and derive equations that describe the effects of BitTorrent's unchoke policy in detail. In Section IV, we use our model to analyze the clustering and data distribution in BitTorrent swarms, and compare this with experimental results published in the literature. In Section V, we validate our model for two classes of peers based on experiments with real BitTorrent clients. Finally, we discuss related work and present our conclusions.

II. THE BITTORRENT PROTOCOL

In this section we provide an overview of the BitTorrent protocol with a detailed description of its unchoke policy.

A. Swarms

In BitTorrent, a *swarm* is a collection of peers that are downloading and/or sharing one particular file. Peers that are in the download process are called *leechers*, while peers that are sharing a complete copy are called *seeders*. Real swarms in the Internet usually contain peers with a variety of upload and download bandwidths.

B. Unchoke policy

Every peer has a number of upload slots available (usually 4-7), and divides its upload bandwidth equally over each of these. Peers that are allocated an upload slot at a peer to download pieces from that peer are called *unchoked* at that peer, while the rest of the peers are *choked*. One of the most crucial aspects of the protocol is the *unchoke policy*, which defines how the upload slots of a peer are allocated to those peers that are interested in pieces from that peer. As numerous BitTorrent clients have been developed over the last decade, implementations of the unchoke policy may vary in some specific details. However, most reliable clients follow the protocol specification of BitTorrent's designer, Bram Cohen, which is implemented in the official *BitTorrent mainline client* [11]. We base our analysis on the protocol followed by mainline client Version 4.0.0 and later. In general, the unchoking of peers is performed in *rounds* of length τ . In practice, τ is usually set to 10 seconds. At each round, the allocation of upload slots is determined. The unchoke policy is different for leechers and seeders.

1) *Leecher unchoke policy*: In every round, a leecher unchokes those interested peers that have recently reciprocated with the highest speed. To do so, it computes a moving average of the rate with which it receives data from all of the peers that upload to it. In addition to this, every three rounds one random interested peer is unchoked as

well, picked in a round-robin fashion from the interested peers. This gives new peers a chance to obtain their first pieces, and provides the unchoking peer with a mechanism to discover faster peers.

2) *Seeder unchoke policy*: As reciprocity is not relevant for the seeders, their unchoke policy is different. In early versions of BitTorrent, seeders simply uploaded to the peers that downloaded the fastest. Since Version 4.0.0 however, a more sophisticated unchoke policy is in place to make the seeding process more 'fair' and to reduce freeriding based on available seeding capacity. In this policy, a certain number ν of random unchokes is spread over three consecutive rounds, again performed in a round-robin fashion. For a particular peer, this number is computed as follows:

$$\nu = \left\lfloor \frac{u+2}{3} \right\rfloor, \quad (1)$$

where u is the number of upload slots of the peer. For example, if $u = 4$, a total of two unchokes are spread over three rounds, i.e., a pattern of 1-1-0 unchokes is followed repeatedly. During each round, the seeder allocates its remaining upload slots in the following order: first, all peers that went from choked to unchoked during the last two rounds remain unchoked; second, the left over slots are allocated to the fastest remaining leechers (based on a moving average of their download rate).

The above policy ensures that every three rounds, ν new peers are unchoked, and that a peer that is unchoked will stay unchoked for at least three rounds.

C. Tracker

Peers in BitTorrent find each other through a central *tracker*, which provides them with a subset of the peers in the swarm. Standard BitTorrent trackers provide this subset at random. In our model we therefore assume that all peers encounter a random and therefore representative selection of other peers.

III. MODEL

In this section we will derive a model of a BitTorrent swarm where peers have different bandwidths. We present a system of differential equations that describes such a swarm in steady state. We then analyze the slot allocations in a system with two classes and extend this analysis to a system with N classes. Finally, we provide an explicit equation for the download speed of peers in such general systems.

A. General approach and assumptions

We follow a similar fluid modeling approach and notation as Qiu *et al.* [7]. In this approach the discrete quantities of numbers of leechers and seeders in a swarm are modeled as continuous variables, and a set of differential equations is derived that describes the dynamics of peers arriving and departing. In steady state, these equations then allow the derivation of the effective download speed of a leecher in the swarm. While the work in [7] is restricted to homogeneous bandwidths, we incorporate the dynamics of having peers with different bandwidths. In our analysis,

we group peers with (roughly) the same bandwidth into a *class*, and we analyze the dynamics of the data exchange within and between classes according to the BitTorrent protocol and its unchoke policy discussed above. Note that our grouping in classes is done for modeling purposes; we do not make any a-priori assumptions about possible biases of peers towards classes. However, later on in our paper we will show that it *emerges* from the BitTorrent protocol that peers mostly unchoke peers with the same bandwidth, a phenomenon that has been called *clustering* in other work, e.g., [9].

Table I contains the notation in our model, which is consistent with the notation in [6], [7]. In our model, every peer has u upload slots. The available upload bandwidth per peer is divided equally over its upload slots, as is the default in standard BitTorrent implementations. We assume that the number of peers per class is large with respect to u . Furthermore, we assume that the download bandwidth of peers is not a bottleneck and that the number of download slots is arbitrary; peers can in principle download pieces from as many other peers they know, as long as these peers have the right pieces. As shown in [7], the efficiency of file sharing in a BitTorrent swarm η (i.e., the likelihood that a peer finds sufficient peers to exchange pieces with) is very close to 1 for files with a high number of pieces. In our model we therefore assume that $\eta = 1$. Additionally, we assume that when a leecher has finished its download, all its upload slots are freed and it ‘re-enters’ the swarm as a seeder for a given period of time. Finally, for convenience of notation, we define $\pi_i(t)$ as the fraction of the leechers that is in class i , i.e.:

$$\pi_i(t) := \frac{x_i(t)}{\sum_j x_j(t)}.$$

B. General fluid model

We assume the peers in a swarm are grouped in N classes, where we can regard the bandwidth of peers in a single class to be the same. The number of downloads completed per second in class i is determined by the total upload bandwidth that class i receives from all classes in the swarm, divided by the file size. We can describe the evolution of $x_i(t)$ and $y_i(t)$ as follows:

$$\begin{aligned} \frac{dx_i}{dt}(t) &= \lambda_i - \frac{\sum_j U_{ji}(t)}{F}, \\ \frac{dy_i}{dt}(t) &= \frac{\sum_j U_{ji}(t)}{F} - \gamma_i y_i(t). \end{aligned} \quad (2)$$

As the total upload bandwidth of class j allocated to class i consists of the upload bandwidth of the seeders and the upload bandwidth of the leechers, we derive $U_{ji}(t)$ as follows:

$$U_{ji}(t) = (\omega_{ji}(t)x_j(t) + \sigma_{ji}(t)y_j(t))\mu_j \quad (3)$$

The core of our multiple class model lies in the derivation of $\omega_{ji}(t)$ and $\sigma_{ji}(t)$, as these depend on the specifics of BitTorrent’s unchoke policy. Before we derive these

μ_i	the upload bandwidth of a peer in class i .
$x_i(t)$	number of leechers in class i at time t .
$y_i(t)$	number of seeders in class i at time t .
λ_i	the arrival rate of leechers in class i .
γ_i	the rate at which seeders in class i leave the system.
Δ_i	the time a seeder in class i stays in the system.
$\omega_{ij}(t)$	the fraction of upload slots of the leechers in class i that is allocated to leechers in class j at time t .
$\sigma_{ij}(t)$	the fraction of upload slots of the seeders in class i that is allocated to leechers in class j at time t .
$U_{ij}(t)$	the total upload bandwidth in class i allocated to leechers in class j at time t .
u	number of upload slots of a peer.
τ	the time interval for optimistic unchoking.
F	the size of the file being downloaded.

Table I
THE NOTATION IN THE MULTIPLE CLASS MODEL.

quantities, we first analyze the properties of a general system in steady state.

C. Steady state analysis

We assume the swarm is in steady state, i.e., while peers are arriving and departing, the total numbers of leechers and seeders in each class are constant. In steady state, it holds that:

$$\frac{dx_i}{dt} = \frac{dy_i}{dt} \equiv 0 \quad (4)$$

We denote the equilibrium values of $x_i(t)$, $y_i(t)$, and $U_{ji}(t)$ by \bar{x}_i , \bar{y}_i , and \bar{U}_{ji} , respectively. Likewise, in the remainder of this paper we denote the equilibrium values of $\sigma_{ij}(t)$, $\omega_{ij}(t)$, and $\pi_i(t)$ by $\bar{\sigma}_{ij}$, $\bar{\omega}_{ij}$, and $\bar{\pi}_i$, respectively. We can reduce the system of Eq. (2) to:

$$\lambda_i F = \sum_j \bar{U}_{ji}, \quad \gamma_i \bar{y}_i F = \sum_j \bar{U}_{ji}. \quad (5)$$

It follows from Eq. (5) that $\bar{y}_i = \lambda_i / \gamma_i$. Furthermore, as the arrival rate of seeders in steady state is equal to the arrival rate of leechers, we can apply Little’s Law to the number of seeders, yielding $\bar{y}_i = \lambda_i \Delta_i$. In a steady state system, it therefore has to hold that $\Delta_i = 1 / \gamma_i$. Combining this with Eqs. (3) and (5) yields the following system:

$$\lambda_i F = \sum_j \left(\bar{\omega}_{ji} \bar{x}_j + \bar{\sigma}_{ji} \lambda_j \Delta_j \right) \mu_j \quad i = 1, \dots, N. \quad (6)$$

Little’s Law applied to the number of leechers yields $\bar{x}_i = \lambda_i T_i$ with T_i the download time of a leecher in class i . In a homogeneous swarm (i.e., with only one class) where all upload bandwidth is utilized, the system of Eqs. (6) reduces to $\lambda F = (\bar{x} + \lambda \Delta) \mu = \lambda (T + \Delta) \mu$. This yields a download time T of:

$$T = \frac{F}{\mu} - \frac{1}{\gamma}, \quad (7)$$

which is consistent with the model in [7] for general file sizes.

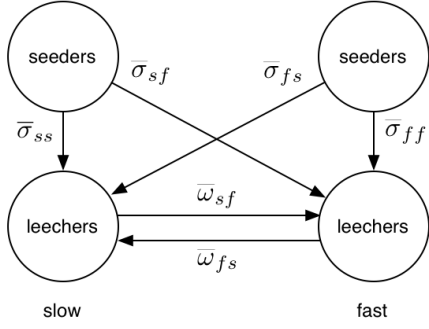


Figure 1. The allocation of upload slots in a swarm with two classes.

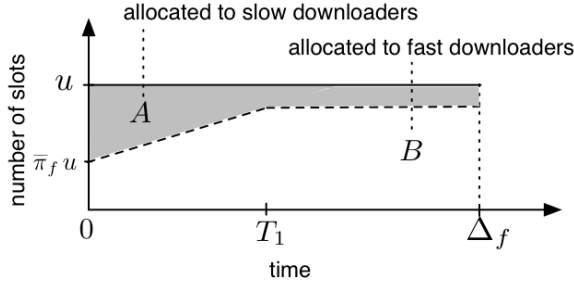


Figure 2. The allocation of upload slots during the lifetime of a fast seeder in a steady state swarm.

D. Two class model

In order to derive the slot fractions $\bar{\sigma}_{ij}$ and $\bar{\omega}_{ij}$ we first analyze the unchoke policy for a swarm with two classes. After this we will generalize our analysis to swarms with N classes.

In the two class model, a swarm consists of a class of slow peers with upload bandwidth μ_s and a class of fast peers with upload bandwidth μ_f , where $\mu_f > \mu_s$. In Figure 1, a schematic overview is given of the distribution of upload slots in such a swarm.

1) *Analysis of seeders:* We first analyze the upload slot allocation of the seeders. As seeders do not depend on reciprocity, at this point in our analysis there is no difference between slow and fast seeders. (Note however that in the overall steady state model, the lifetime of slow seeders can be different from that of fast seeders, and a distinction should be made.) At points of choice we arbitrarily use the case of a fast seeder in our analysis.

We define $S_{ff}(t)$ as the number of slots a seeder has allocated to fast peers at time t . First, we look at the slot allocation right after the arrival of the seeder at $t = 0$. When the seeder has just arrived, it does not yet have information about the speed of other peers; it only acquires this information after it has unchoked some random peers for some time. Therefore, the slot allocation at $t = 0$ is fully random. Stochastically, this implies that $S_{ff}(0) = \bar{\pi}_f u$.

On the other hand, when the seeder is long enough in the system to know enough fast peers, it will hold that $S_{ff}(t) = (u - \nu) + \bar{\pi}_f \nu = u - \bar{\pi}_s \nu$. This follows from

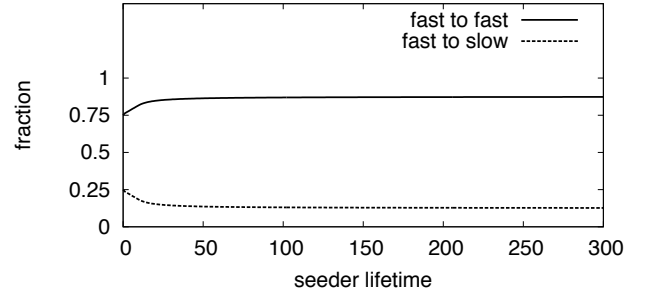


Figure 3. The average fraction of upload slots $\bar{\sigma}_{ff} = \bar{\sigma}_{sf}$ of a seeder allocated to fast and slow leechers in a steady state swarm.

the unchoke policy explained in Section II; during a period of three rounds, exactly ν random peers are unchoked and the remaining slots are allocated to the fastest downloaders. We define the time at which this state is reached as $t = T_1$.

The slot allocation between $t = 0$ and $t = T_1$ can be understood in the following way. As ν ‘new’ peers are ‘tried out’ every three rounds, on average $\bar{\pi}_f \nu / 3$ fast peers are discovered per round. Therefore, the $S_{ff}(t)$ will increase linearly from $S_{ff}(0)$ to $S_{ff}(T_1)$. This is visualized in Figure 2. We can express the number of slots $S_{ff}(t)$ as:

$$S_{ff}(t) = \min\left(\bar{\pi}_f u + \frac{\bar{\pi}_f \nu t}{3}, u - \bar{\pi}_s \nu\right). \quad (8)$$

It follows that:

$$T_1 = 3\left(\frac{u - \nu}{\bar{\pi}_f \nu} + \frac{\nu - u}{\nu}\right). \quad (9)$$

In our model, we are interested in the overall fraction of upload slots all seeders combined have allocated to fast, respectively, slow leechers in steady state. In steady state, the ages of the seeders in the system are uniformly distributed over the time interval $[0, \Delta_f]$. Therefore we can compute the fraction of upload slots allocated to fast leechers by computing the relative area of B in Figure 2:

$$\bar{\sigma}_{ff} = \bar{\sigma}_{sf} = \frac{B}{A + B} = \frac{1}{\Delta_f u} \int_0^{\Delta_f} S_{ff}(t) dt. \quad (10)$$

Obviously, $S_{fs} + S_{ff} = u$ and $\bar{\sigma}_{fs} = 1 - \bar{\sigma}_{ff}$. Summarizing, using Eq. (10) the distribution of upload slots of the seeders over the two classes can explicitly be computed. As an example, in Figure 3 we have plotted the value of $\bar{\sigma}_{ff}$ for different values of Δ_f in a swarm where 75% of the peers are fast with $u = 4$ and $\tau = 10$. Clearly, $\bar{\sigma}_{ff}$ converges to $(u - \bar{\pi}_s \nu) / u$, which is 0.875 in this case.

2) *Analysis of leechers:* The fast leechers stay in the system as a leecher until their download is finished. We denote the download time of the fast leechers by T_f . Due to the ‘tit-for-tat’ policy, a leecher only continues uploading to another leecher as long as it can continue downloading from that leecher. The upload slot allocation of a fast leecher $\bar{\omega}_{ff}$ can therefore be derived in a similar way as that of the seeders. In the case of a leecher, only one random peer is unchoked every three rounds, and so, analogous to Eq.

(8), we can derive the number of upload slots allocated by fast leechers to fast leechers D_{ff} as:

$$D_{ff}(t) = \min\left(\bar{\pi}_f u + \frac{1}{3}\bar{\pi}_f \frac{t}{\tau}, u - \bar{\pi}_s\right). \quad (11)$$

Consequently, the fraction $\bar{\omega}_{ff}$ in steady state can be derived as:

$$\bar{\omega}_{ff} = \frac{1}{T_{fu}} \int_0^{T_f} D_{ff}(t) dt. \quad (12)$$

Just as with the seeders, it is obvious that $D_{fs}(t) + D_{ff}(t) = u$ and $\bar{\omega}_{fs} = 1 - \bar{\omega}_{ff}$.

Finally, we analyze the slot allocation of the slow leechers. In total, $\bar{x}_f D_{fs}(t)$ slots are allocated from the fast peers to the \bar{x}_s slow peers. Due to the tit-for-tat policy, for every slot a fast peer allocates to a slow peer, a slow peer allocates a slot to a fast peer (while not vice versa). In addition, a slow peer also optimistically unchokes fast peers. This implies that:

$$D_{sf}(t) = \frac{\bar{x}_f}{\bar{x}_s} D_{fs}(t) + \bar{\pi}_f. \quad (13)$$

Note that when the slot allocation has stabilized it simply follows that $D_{fs}(t) = \bar{\pi}_s$ and hence $D_{sf}(t) = 2\bar{\pi}_f$.

Based on the above, $D_{ss}(t)$, $\bar{\omega}_{sf}$, and $\bar{\omega}_{ss}$ are known as well.

E. N class model

Using the same approach as above, the model can be extended for a general swarm with N classes.

We assume that there are N classes of peers in the swarm with upload bandwidths μ_1, \dots, μ_N . We order the classes in such a way that $\mu_i < \mu_j$ if and only if $i < j$. We first analyze the number of slots a seeder in class i allocates to leechers of class j , denoted by $S_{ij}(t)$. Analogously to the two-class case, it holds stochastically that $S_{ij}(0) = \bar{\pi}_j u$.

Furthermore, as the peers in class N are the fastest peers, there is a time T_N such that for $t \geq T_N$:

$$\begin{aligned} S_{ij}(t) &= \bar{\pi}_j \nu, \\ S_{iN}(t) &= u - \nu + \bar{\pi}_N \nu. \end{aligned} \quad (14)$$

Analogously to the two class model, it follows that:

$$\bar{\sigma}_{ij} = \frac{1}{\Delta_i u} \int_0^{\Delta_i} S_{ij}(t) dt. \quad (15)$$

We will now analyze the slot allocation between $t = 0$ and $t = T_N$. Over every period of three rounds, a seeder unchokes the $u - \nu$ fastest peers it knows. However, it only knows the speed of a peer after this peer has been randomly unchoked for some time. Therefore, at a certain time $t \in (0, T_N)$, the number of slots allocated to class i depends on the number of *faster* peers that have been ‘discovered’ with the random unchokes before t .

We will illustrate the resulting slot allocation with an example of three classes: fast, medium, and slow. Let us assume that the proportion of the sizes of these respective classes is 1:4:5. It follows that a random unchoke has a much higher chance to yield a medium peer, than to yield a fast peer. Therefore, the $u - \nu$ slots at first rapidly fill with

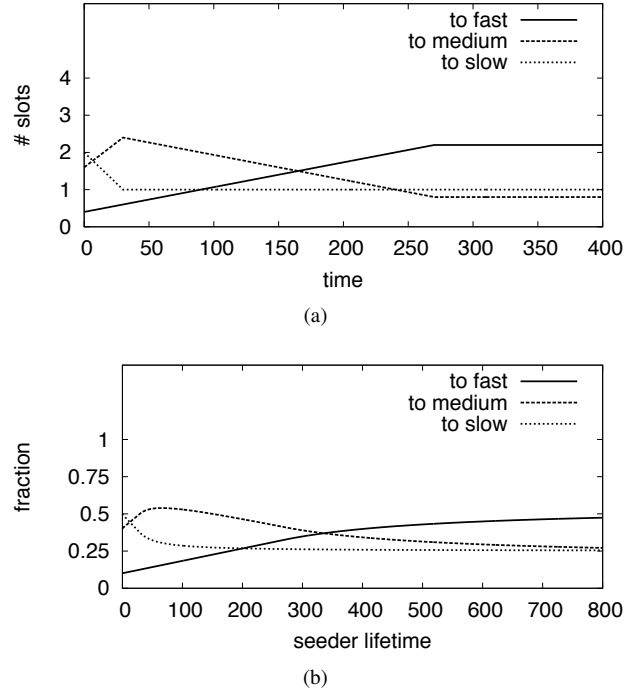


Figure 4. (a) The allocation of upload slots $S_{ij}(t)$ of a seeder in an arbitrary class to leechers in slow, medium, and fast classes; (b) The fraction of slots $\bar{\sigma}_{ij}$ a seeder has allocated to the three classes, for various lifetimes Δ_i .

medium peers, and only slowly fill with fast peers. This proceeds until all $u - \nu$ slots are filled by fast peers (which happens at T_N mentioned above). As a result, the average number of slots allocated to medium peers first increases, then decreases, and then reaches a fixed value. In Figure 4a we have plotted the number of slots $S_{ij}(t)$ a seeder has allocated to each of the three classes; in Figure 4b the corresponding fraction of slots $\bar{\sigma}_{ij}$ for various lifetimes Δ_i .

The derivation of the slot allocation of leechers $D_{ij}(t)$ and $\bar{\omega}_{ij}$ proceeds along the same lines as in the two-class case. Just as with the seeders, after a certain time the slot allocation will become stable. Due to tit-for-tat and optimistic unchoking it holds that:

$$D_{ij}(t) = \frac{\bar{x}_j}{\bar{x}_i} D_{ji}(t) + \bar{\pi}_j, \quad i < j. \quad (16)$$

The allocation of a peer's slots to equal or slower peers depends on the number of slots not already allocated to faster peers or by random unchoking to slower peers. After the stabilization of the slot allocation, it therefore holds that:

$$D_{ij}(t) = u - \sum_{k < j} \bar{\pi}_k - \sum_{k > j} D_{ik}(t), \quad i \geq j. \quad (17)$$

Eq. (17) can be solved inductively, starting with $D_{NN} = u - 1 + \bar{\pi}_N$. For general i, j , this yields:

$$\begin{aligned} D_{ij}(t) &= 2\bar{\pi}_j, & j > i \\ D_{ij}(t) &= \bar{\pi}_j, & j < i \\ D_{ii}(t) &= u - 1 + \bar{\pi}_i - \sum_{k > i} \bar{\pi}_k. \end{aligned} \quad (18)$$

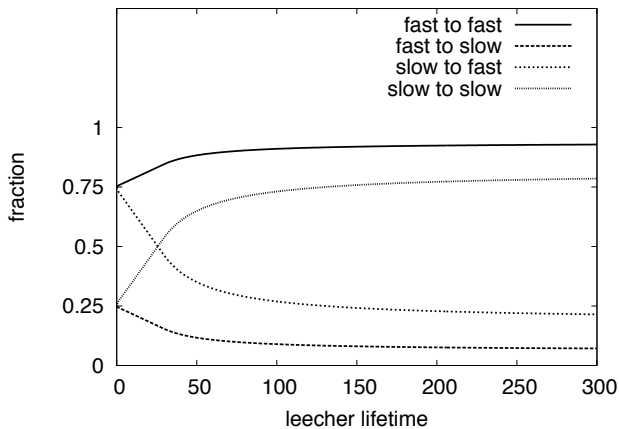


Figure 5. The average fractions of upload slots $\bar{\omega}_{ff}, \bar{\omega}_{fs}, \bar{\omega}_{sf}, \bar{\omega}_{ss}$ of fast, respectively, slow leechers allocated to fast, respectively, slow leechers in a steady state swarm with 75% fast and 25% slow peers.

When the values of $D_{ij}(t)$ are known, the fractions $\bar{\omega}_{ij}$ can be computed analogously to Eq. (12):

$$\bar{\omega}_{ij} = \frac{1}{T_i u} \int_0^{T_i} D_{ij}(t) dt. \quad (19)$$

F. Download speed

If the values of \bar{x}_i and \bar{y}_i are explicitly known, and if for simplicity we neglect the stabilization period right after the arrival of a peer, the download speed of a peer in class i is given by:

$$d_i = \frac{1}{\bar{x}_i u} \left(\sum_j (D_{ji} \bar{x}_j + S_{ji} \bar{y}_j) \mu_j \right) \quad (20)$$

For any given swarm, the required values for D_{ij} and S_{ij} can be easily determined using Eqs. (14) and (18). Hence, we have hereby presented a method with which the download speeds in any BitTorrent swarm with N classes can be computed straightforwardly.

G. Solving the system when only arrival rates are known

In case the values of \bar{x}_i and \bar{y}_i are not explicitly known, these can be computed by solving Eq. (6) and substituting Eqs. (15) and (19) for $\bar{\sigma}_{ij}$ and $\bar{\omega}_{ij}$. This yields a system of N equations with N unknowns. The solutions can be computed trivially using a computer. Depending on the actual values of the various parameters (e.g., λ_i , Δ_i , etc.) some of the solutions can have negative values for one or more \bar{x}_i . Obviously, these solutions correspond to steady states that are not feasible in practice. In the case where no feasible solution exist, a real system with the given parameters would not reach a steady state.

IV. CLUSTERING AND DATA DISTRIBUTION ANALYSIS

We will now use our model to analyze bandwidth clustering and data distribution in BitTorrent swarms.

A. Bandwidth clustering

Legout *et al.* [9] presented very interesting results that indicated an emergent clustering in BitTorrent swarms where peers “have a clear preference for peers with similar upload capacities”. However, to understand the BitTorrent protocol properly, it is essential to realize that the clustering only implicitly follows from the unchoke policy. In fact, *all* peers prefer to upload to the fastest peers possible, as long as these reciprocate. Since the fastest peers only allocate a few slots to slower peers, the slower peers only allocate a few slots in return to faster peers. Hence, the slower peers resort to allocating a large part of their slots to slower peers. As a result, peers group around bandwidth.

Our model confirms the above effect. In Figure 5, the slot allocation is displayed for a swarm with 75% fast peers and 25% slow peers based on Eqs. (11), (12), and (13). The fast peers clearly allocate most of their upload slots to fast peers, while the slow peers *as a result* allocate most of their upload slots to slow peers.

We have also used our model to explain the experimental results of [9] for their main test scenario with three classes. In this test scenario, a file of 113 MB is downloaded in a swarm with peers of three different upload bandwidths: peers 1 to 13 have a 20 KB/s upload bandwidth, peers 14 to 26 have a 50 KB/s upload bandwidth, peers 27 to 39 have a 200 KB/s upload bandwidth. In Figure 6a, the average slot allocation of each peer to each other peer as computed by our model is displayed. The allocation of a single peer to a class is averaged over all peers in that class. The grouping of peers with the same upload bandwidths is clearly visible. In [9], the total *number* of unchokes during the lifetime of the peer is displayed, which is obviously larger for the slower peers than for the faster peers. In order to allow comparison with their experiment, we normalized the slot allocation with respect to the lifetime of the peers in Figure 6b. Just as in [9], a very interesting block-wise allocation is visible and our model confirms their experimental results.

B. Data distribution

In Figure 7 we have displayed how the incoming data of a peer originates from the different classes in the system, as well as the peer’s slot allocation *to* these different classes. In this scenario, fast peers have an upload bandwidth of 1024 KB/s and slow peers have an upload bandwidth of 128 KB/s. We assessed swarms with various proportions of fast peers versus slow peers. In all cases, there are twice as many leechers as there are seeders. It is interesting to observe how much data the slow peers receive from fast peers, despite the fact that they only unchoke a very limited fraction of their upload slots to the fast peers. Therefore, while peers clearly cluster in terms of how they allocate *upload slots* to each other, the effect of this on the actual amount of data transferred in the system is less dominant than what would seem at first sight. When 90% of the peers is fast, even though the fast peers only unchoke a tiny fraction of their upload slots to slow peers (Fig. 7a), they account for over 75% of the bandwidth received by slow peers (Fig. 7b). Even when only 10% of the peers is

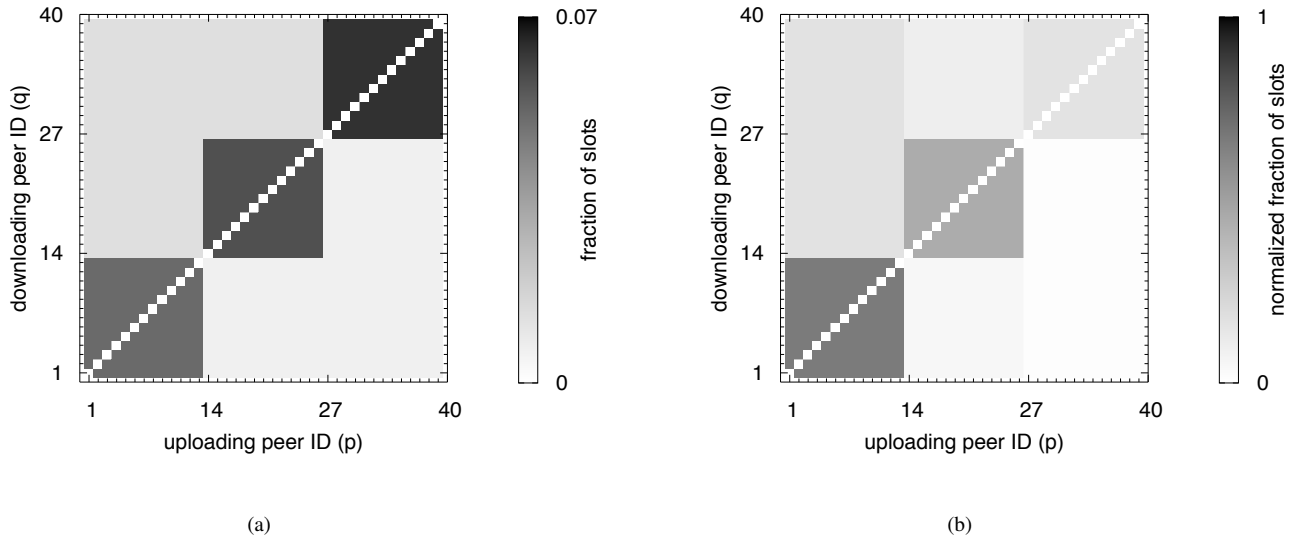


Figure 6. (a) The fraction of its upload slots peer p allocates on average to peer q according to the model; (b) The same fraction of slots, normalized to the download time of peer p . A darker square indicates a higher fraction of slots. Peers 1 to 13 have a 20 KB/s upload bandwidth, peers 14 to 26 have a 50 KB/s upload bandwidth, peers 27 to 39 have a 200 KB/s upload bandwidth.

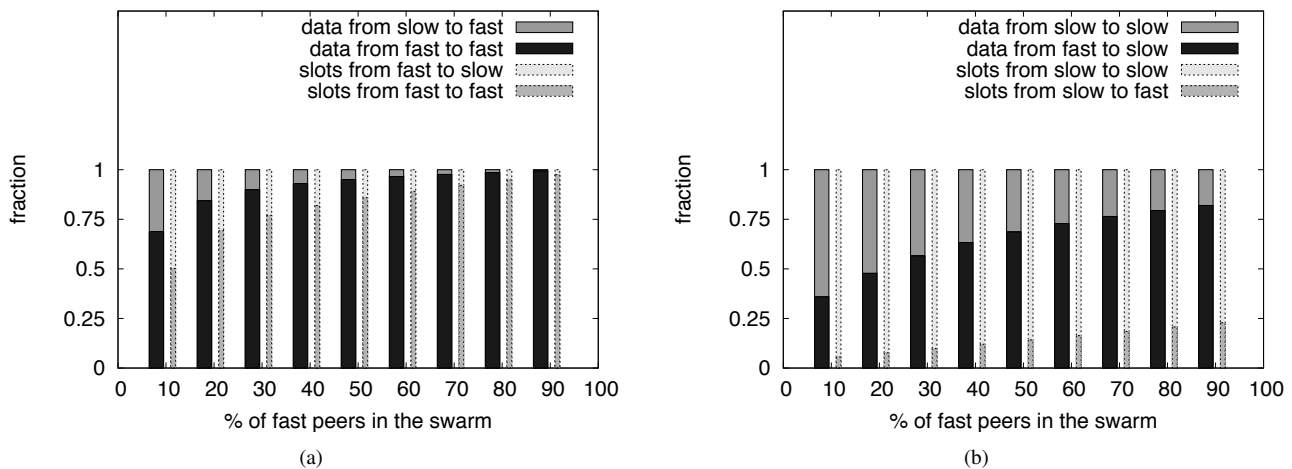


Figure 7. The distribution of the data that a peer receives *from* different classes, as well as the fraction of slots this peer allocates *to* these classes, for (a) a fast peer and (b) a slow peer, in swarms with different percentages of fast peers. We assume that a peers' download bandwidth is not a bottleneck.

fast, they still account for more than 30% of the bandwidth received by slow peers.

Summarizing, we conclude that a strong clustering indeed emerges in BitTorrent swarms both from theory and from practice. However, for understanding the actual data distribution within the system this clustering effect can be misleading, and the actual bandwidth distribution of peers is of crucial importance.

V. EXPERIMENTS

In this section we will describe the experiments with which we have validated our model for two classes. We have used real BitTorrent clients and compare our model with the experimental results of various swarm configurations.

A. Experimental setup

In order to obtain realistic results that are based on the actual BitTorrent protocol, we have set up an experimental environment based on running real instances of the BitTorrent mainline client [11]. We used Version 4.4.0 of this client, which implements the policies as described in Section II. In order to have full control over a swarm, we have set up a local tracker and used a custom made file of 100 MB with its associated torrent. In each of our experiments, we execute a number of clients on a single testing machine in various class configurations and monitor their unchoke behavior and piece exchanges. In order not to be constrained by disk writing bottlenecks, we slightly altered the client code in such a way that no data are written to disk or read from disk while downloading the file.

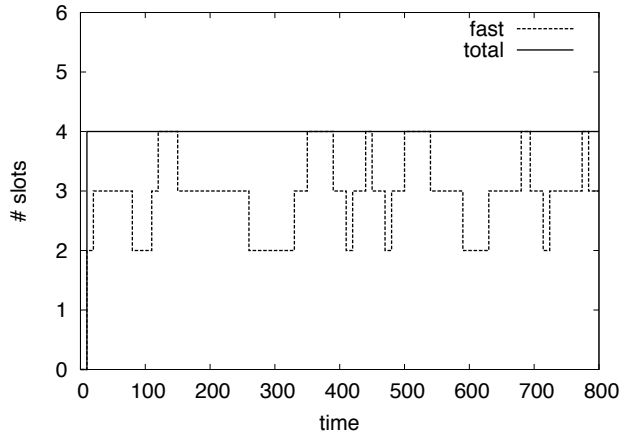


Figure 8. The upload slot allocation of a seeder in a swarm with 50% slow peers and 50% fast peers.

% fast peers	$\bar{\sigma}_{ff} = \bar{\sigma}_{sf}$	
	model	exp
30	0.65	0.6368
50	0.75	0.7443
70	0.85	0.8486

Table II

MODEL PREDICTIONS VERSUS EXPERIMENTAL VALUES OF THE AVERAGE FRACTION OF SLOTS ALLOCATED TO FAST PEERS FOR A SEEDER IN SWARMS WITH DIFFERENT FRACTIONS OF FAST PEERS.

With the resulting setup, it is feasible to run swarms of up to 50 peers on our single test machine. We are planning to extend our test environment and implement it on a multi-processor supercomputer, so as to obtain results for larger swarms with more than two classes.

In all of our experiments, slow peers have an upload and download bandwidth of 5 KBps, while fast peers have an upload and download bandwidth of 200 KBps. These values are low, but allow for very precise measurements and prevent memory speed from becoming a bottleneck on our testing machine. Furthermore, we set the number of upload slots per peer to 4 and the duration of a round to the standard value of 10 seconds.

B. Behavior of the seeders

We first validate the model for the unchoking behavior of seeders, according to the policy as implemented in the mainline client. As mentioned before, a seeder does not take into account reciprocity. As a consequence, the unchoking behavior of a seeder is independent of its upload and download bandwidth. In our experiments, we have executed different swarm configurations consisting of various fractions of slow and fast leechers. We then executed a single seeder and monitored its unchoking and uploading behavior.

Figure 8 displays the number of slots unchoked during the seeder's lifetime, as well as the slots unchoked specifically to fast leechers for a swarm configuration of 50% slow peers and 50% fast peers. The effect of the protocol as described in Section II is clearly visible as a certain number of fast peers is always unchoked (here 2) while over the

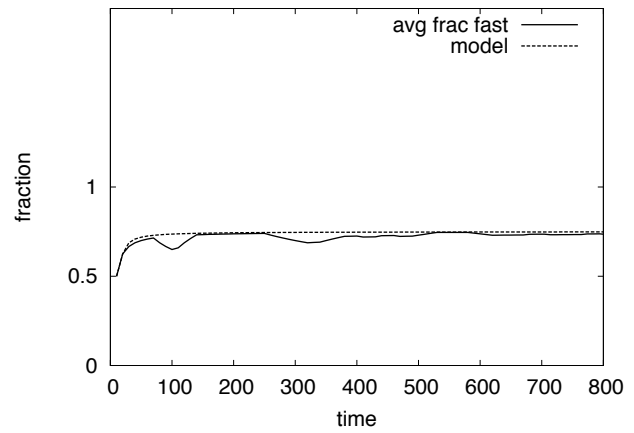


Figure 9. The average fraction $\bar{\sigma}_{ff}$ of upload slots of a seeder allocated to fast leechers over its lifetime.

remaining slots a round-robin-like random pattern emerges. It follows from Eq. (1) with $u = 4$ that ν should be equal to 2. This is also visible in the figure, as the random unchokes proceed over exactly two slots.

In Figure 9, the average fraction of slots unchoked to fast peers over the lifetime of the seeder is plotted, together with the prediction by our model. The actual measured value stays very close to the predicted value and after some time hardly fluctuates anymore.

We performed the above experiment also for other configurations of the two classes. In Table II, the model predictions as well as the converged experimental values are displayed for fractions of 30%, 50%, and 70% fast leechers in the swarm. For all configurations the deviation from the model is minimal.

C. Behavior of the leechers

The unchoking behavior of a leecher is more complicated, as it depends on reciprocity and is therefore dependent on the behavior of other peers, the piece transfers that are occurring at a specific moment in time, and the availability of pieces at other peers. In our experiments, we created a steady state with specific fractions of slow leechers and fast leechers. We monitored the unchoking behavior of both classes of leechers for all of the leechers in the swarm. As a specific example of leecher behavior, Figure 10 displays the slot unchoking of a single fast leecher in a swarm of 50% slow peers and 50% fast peers. Right after the peer's arrival, not all slots are unchoked. This is due to the fact that during the first few rounds a peer has (almost) no pieces to exchange. After some time all slots are allocated and the identified pattern becomes visible. Note however that this is just a single leecher instance. In order to properly validate our model predictions, we averaged the fractions of unchoked fast peers over all the monitored slow, respectively, fast leechers. Figures 11 and 12 display the average fraction of slots $\bar{\omega}_{ff}$ of a fast leecher, respectively, a slow leecher unchoked to fast leechers. In the early stages of the download, there are some fluctuations, likely due to specific piece exchange possibilities. After some time, the

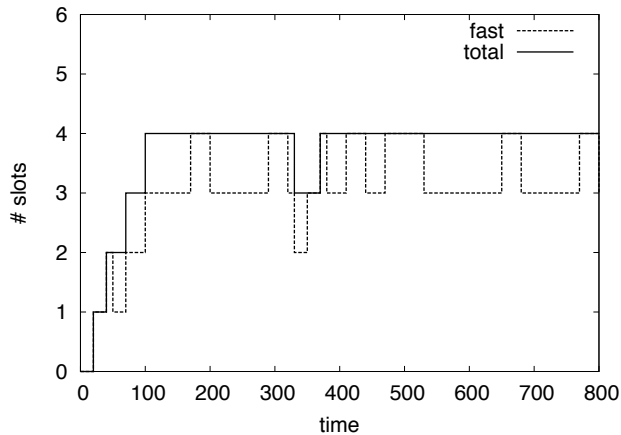


Figure 10. The upload slot allocation of a particular leecher in a swarm with 50% slow peers and 50% fast peers.

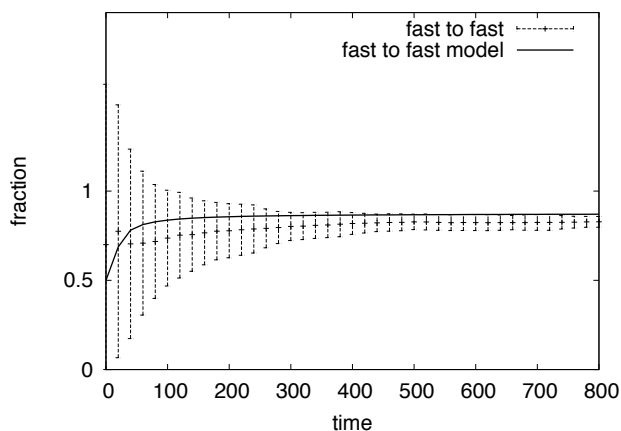


Figure 11. The average fraction $\bar{\omega}_{ff}$ of upload slots of fast leechers allocated to fast leechers over their lifetime. The data is plotted with a 95% confidence interval.

measured value in both cases quite accurately converges to the predicted value. For $\bar{\omega}_{ff}$ however, the converged value stays a little under the model prediction.

In Table III the results are displayed for fractions of 30%, 50%, and 70% fast leechers. It is visible that for higher fractions of fast peers, our model seems more accurate. For lower fractions of fast peers, the model predicts slightly higher values for $\bar{\omega}_{ff}$ and slightly lower values for $\bar{\omega}_{sf}$. We suspect that these differences are caused by probabilistic effects that arise due to the limited scale of the swarms in our experiments. When a certain class i has a small number of peers, the unchoke of a number of peers from that class significantly influences the number of remaining peers in that class, and therefore affects the probability distribution of a random unchoke in the future. For larger classes this influence becomes negligible. Hence, in future research we plan to perform large scale experiments on a parallel-processor machine to further investigate this effect.

VI. RELATED WORK

A variety of generic and specific P2P models has been proposed in earlier work. Gaeta *et al.* [3] provide a stochas-

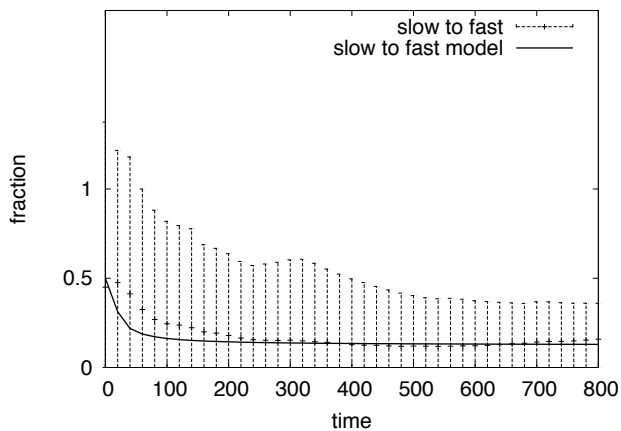


Figure 12. The average fraction $\bar{\omega}_{sf}$ of upload slots of slow leechers allocated to fast leechers over their lifetime. The data is plotted with a 95% confidence interval.

% fast peers	$\bar{\omega}_{ff}$		$\bar{\omega}_{sf}$	
	model	exp	model	exp
30	0.825	0.7612	0.075	0.1186
50	0.875	0.8349	0.125	0.1284
70	0.925	0.9040	0.175	0.1694

Table III
MODEL PREDICTIONS VERSUS EXPERIMENTAL VALUES OF $\bar{\omega}_{ff}$ AND $\bar{\omega}_{sf}$ FOR DIFFERENT FRACTIONS OF FAST PEERS.

tic fluid model for general P2P resource exchanges, but do not consider BitTorrent-like tit-for-tat mechanisms. Kumar *et al.* [2] derive the minimum distribution time in systems with equal peers, and in systems where one homogeneous set of peers explicitly forwards bandwidth to another homogeneous set of peers (but not vice-versa). Munding *et al.* [4] introduce a generic model with different upload capacities of peers and present a mixed integer linear program to determine the minimum distribution time, but they do not consider tit-for-tat mechanisms. DeFigueiredo *et al.* [5] provide performance bounds for the tit-for-tat strategy in homogeneous systems. They assess the influence of cooperation in such systems and argue that there is a trade-off with tit-for-tat between fairness and efficiency. Qiu *et al.* [7] provide a fluid-based model of BitTorrent for homogeneous bandwidth configurations, which we have extended in this paper for inhomogeneous swarms. Furthermore, they analyze the piece distribution efficiency and recommend certain improvements to the BitTorrent protocol. Parvez *et al.* [6] extend the homogeneous model of [7] for analysis of on-demand media streaming. Xiangying *et al.* [12] assess the service capacity and role of resource policies in a similar way based on homogeneous upload capacities. The authors focus on a dynamic system and analyze both the transient phase after the introduction of a file, and the stationary phase when the performance has more or less stabilized. Legout *et al.* [9] provide empirical evidence of bandwidth clustering in BitTorrent swarms. We have compared their results with the predictions of our model in Section IV. Finally, Bendadis *et al.* [1] analyze bandwidth conservation

in systems under various rate requirements such as Video-on-Demand. An equation for the performance in a tit-for-tat system is provided under certain limited assumptions, but further analysis and solutions are omitted. None of all these publications provide an accurate model for BitTorrent swarms with arbitrary bandwidth configurations.

VII. CONCLUSIONS

In this paper, we have provided a generic mathematical model of bandwidth-inhomogeneous BitTorrent swarms, based on a detailed analysis of BitTorrent's unchoke policy for swarms which consist of N classes of peers with different bandwidths. We used our model to analyze clustering and data distribution in BitTorrent swarms, and showed that our model accurately predicts the experimental evidence of bandwidth clustering presented in other work. However, we showed that this grouping of peers with the same bandwidth might be misleading in understanding the actual data distribution in the swarm. In our scenarios, slow peers mostly unchoked other slow peers, but received most of their data from fast peers. Furthermore, we validated our model experimentally for swarms with two classes by running controlled swarms of BitTorrent clients. In future work, we plan to perform large scale experiments with more than two classes to gain more insight into the properties of bandwidth-inhomogeneous swarms.

ACKNOWLEDGEMENTS

This research was supported by the EU FP6 project P2P-FUSION with project reference 035249 and by the EU FP7 project P2P-NEXT with project reference 216217.

REFERENCES

- [1] F. Benbadis, F. Mathieu, N. Hegde, and D. Perino, "Playing with the bandwidth conservation law," in *P2P '08: Proceedings of the Eighth IEEE International Conference on Peer-to-Peer Computing*, 2008, pp. 140–149.
- [2] R. Kumar and K. Ross, "Optimal peer-assisted file distribution: Single and multi-class problems," in *Proc. of IEEE Workshop on Hot Topics in Web Systems and Technologies (HOTWEB'06)*, July 2006.
- [3] R. Gaeta, M. Griboudo, D. Manini, and M. Sereno, "Analysis of resource transfers in peer-to-peer file sharing applications using fluid models," *Performance Evaluation*, vol. 63, no. 3, pp. 149–174, 2006.
- [4] J. Munding, R. R. Weber, and G. Weiss, "Analysis of peer-to-peer file dissemination amongst users of different upload capacities," in *SIGMETRICS'06: Proceedings of the 2006 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM Press, 2006, pp. 5–6.
- [5] D. DeFigueiredo, B. Venkatachalam, and S. F. Wu, "Bounds on the performance of p2p networks using tit-for-tat strategies," in *P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 11–18.
- [6] N. Parvez, C. Williamson, A. Mahanti, and N. Carlson, "Analysis of bittorrent-like protocols for on-demand stored media streaming," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 301–312, 2008.
- [7] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *ACM SIGCOMM*, Portland, OR, USA, August 2004.
- [8] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, USA, May 2003, <http://bittorrent.com>.
- [9] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in bittorrent systems," in *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM Press, 2007, pp. 301–312.
- [10] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent?" in *NSDI'07*, Cambridge, MA, April 2007.
- [11] "Bittorrent mainline client," <http://www.bittorrent.com>.
- [12] X. Yang and G. de Veciana, "Performance of peer-to-peer networks: service capacity and role of resource sharing policies," *Performance Evaluation*, vol. 63, no. 3, pp. 175–194, 2006.