Peer-Assisted File Distribution: The Minimum Distribution Time

Rakesh Kumar
Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY
Email: rkumar04@utopia.poly.edu

Keith W. Ross
Department of Computer and Information Science, Polytechnic University, Brooklyn, NY
Email: ross@poly.edu

***Abstract*— With the emergence of BitTorrent, Swarmcast, and CDNs, peer-assisted file distribution has become a prominent Internet application, both in terms of user popularity and traffic volumes. We consider the following fundamental problem for peer-assisted file distribution. There are seed nodes, each of which has a copy of the file, and leecher nodes, each of which wants a copy the file. The goal is to distribute the file to all the leechers – with the assistance of the upload capacity of the leechers – in order to minimize the time to get the file to all the leechers (the distribution time). We obtain explicit expressions for the minimum distribution time of a general heterogeneous peer-assisted file distribution system. Derived with fluid-flow arguments, the expressions are in terms of the file size, the seeds' upload rates and the leechers' upload and download rates. We demonstrate the utility of the result by comparing the optimal distribution time with the measured distribution time when BitTorrent is used to distribute a file from a seed to ten leechers.**

I. Introduction

In classic client/server file distribution, a set of servers distributes a file to receiving nodes. This file could be a software distribution, a patch, or content such as a movie, album, or a TV show. If the file size and the number of receiving nodes is large, the servers and the servers' bandwidth can become bottlenecks in the distribution process.

Peer-assisted file distribution leverages the uploading capacity of the receiving nodes (peers) to aid in the file distribution process. Specifically, once a node has received any portion of the file, it can redistribute that portion to any of the other receiving nodes. Today, there are numerous examples of peer-assisted file distribution, including:

- **File Distribution to CDN Servers:** CDNs typically include a large number of servers. When a CDN is charged with the distribution of new content, the CDN must push the content from a small number of source nodes to all (or many) of its servers. Once a server has received a portion of the file, it can replicate and send the portion to other CDN servers, thereby diminishing the burden on the source nodes. (After the file has been pushed to all the servers, clients can then pull the file from the CDN servers.)
- **BitTorrent:** BitTorrent [9], [15], [18] is a popular peer-assisted file distribution application, currently accounting for a significant portion of Internet traffic [6]. BitTorrent is used for Linux distributions as well as for distribution of multimedia content. In BitTorrent, the servers are referred to as seeds and the receiving nodes are referred to as leechers. In the same spirit, numerous other file distribution protocols have also been proposed, including Slurpie [19], Swarmcast [20], and Avalanche [1], [11].
- **SplitStream:** Splitstream [7] stripes the file, and distributes the stripes using separate application-level multicast trees with disjoint interior nodes.

The inherent scalability of these protocols permits distribution of large size files to several thousand users without demanding excessive bandwidth usage at the distribution servers. This way an ordinary PC with cable/DSL connection can be used to distribute large files to an audience of size orders of magnitude higher than what was possible with the classic client-server based distribution techniques.

Clearly, peer-assisted file distribution has become an important application paradigm in the Internet. But, quantitatively, just how good is it at distributing a file? Can it be significantly better than client/server distribution? How well can it scale as the number of receiving nodes becomes very large? How does the interaction of server upload bandwidth, receiving node upload band-

width, and receiving node download bandwidth impact the overall distribution time?

In this paper, we address fundamental questions like these lying at the core of peer-assisted file distribution. Using fluid arguments, we first derive an expression for the minimum achievable file distribution time in terms of the basic parameters of a peer-assisted file distribution system, namely, the file size, the number of servers, the number of receiving nodes, and the upload and download bandwidths of all the participating nodes. The expression is general in the sense that it accounts for arbitrary and heterogeneous upload and download rates. Moreover, the expression is in closed form and is remarkably simple. We then use this result to address many of the questions posed above.

It is important to note that the optimal peer-assisted distribution problem is not a variation of the max-flow problem [10] from graph theory. In a classical graph-theory flow problem, the rate at which fluid flows out of a node cannot exceed the rate at which fluid flows into the node. However, in file distribution, owing to the possibility of replication, the rate of bits leaving a node may exceed the rate at which the bits enter the node. Thus, the network flow problems from graph theory have little bearing on the problem at hand.

After deriving the optimal distribution time, we apply the theory to a real P2P content-distribution scheme. Specifically, we now examine how Azureus [2], a popular BitTorrent client stacks up against the optimal distribution. To this end, we use Azureus to distribute a file from 1 seed host to 10 leecher hosts in a closed environment, measure the distribution time, and compare the distribution time to the optimal distribution time. For this modest-size network, it is shown that Azureus is fairly efficient when either the minimal download rate or the aggregate upload rates is the bottleneck; however, Azureus is less efficient when the servers' aggregate upload rate is the bottleneck.

We mention here that there has been recent related work addressing the distribution time in peer-assisted file distribution [5], [8], [17]. However, the contribution of these works has been limited to specific homogeneous models and, in some papers, to specific overlay topologies. Furthermore, none of these related works address peer-assisted distribution with differentiated services. To our knowledge, this paper is the first to determine the minimum attainable distribution time for a general heterogenous system. We discuss the related work at the end of this paper in Section VI.

The rest of the paper is organized as follows. In Section II we formally describe the problem and explain the possible limitations in applicability of the results derived in the paper. In Section III we present our main result for the single-class case and in Section IV we discuss the related insights. In Section V we use our results to benchmark the Azureus implementation of BitTorrent in a measurement study. Section VI describes the related work and we conclude in Section VII.

II. Problem Description

We consider the following fundamental problem in file distribution. There are two sets of nodes: the *seeds* denoted by $\mathcal{S}$; and the *leechers* denoted by $\mathcal{L}$. Each seed has a copy of a file of size F, and each leecher in $\mathcal{L}$ desires a copy of the file. Initially none of the leechers has any portion of the file, but over time the leechers obtain portions. A leecher can obtain portions of the file from any of the seeds and from other leechers that have portions. A leecher is permitted to leave after obtaining the entire file. Let $\mathcal{I} = \mathcal{S} \cup \mathcal{L}$ be the set of all nodes.

Each node (seed or leecher) i has an upload bandwidth u_i; and each leecher has a download bandwidth d_i. A node i can transmit bits at a maximum rate of u_i and can download bits at a maximum rate of d_i. Today in the Internet, typically $u_i \leq d_i$; however, we allow for arbitrary upload and download rates.

Consider the problem of distributing the file, from seeds to leechers and among leechers, to minimize the *distribution time*, that is, the time it takes to get the entire file to all of the leechers. To be more precise, denote the rate at which leecher $i \in \mathcal{L}$ downloads 'fresh' content from seeds and other leechers combined at time t by $r_i(t)$. We refer to $\{r_i(t),\ t \geq 0,\ i \in \mathcal{L}\}$ as a rate profile. For each rate profile $\{r_i(t),\ t \geq 0,\ i \in \mathcal{L}\}$, there is a distribution time. Denote by $T_{\min}$ as the minimum distribution time achievable over all possible rate profiles.

The first goal of this paper is to determine the minimum distribution time $T_{\min}$ and the corresponding rate profile $\{r_i(t),\ t \geq 0,\ i \in \mathcal{L}\}$ which achieves this $T_{\min}$ for arbitrary values of F, u_i, $i \in \mathcal{I}$, and d_i, $i \in \mathcal{L}$. In many cases there are many different rate profiles that achieve $T_{\min}$; we are not so much interested in characterizing the set of optimal rate profiles as we are in determining $T_{\min}$. However, as mentioned before as a by-product of our analysis, we will determine an optimal rate profile for each of the considered cases.

Our model is essentially a fluid model with fluid replication at leecher nodes. In particular, we assume that a leecher node can replicate and forward a bit as soon

as it receives the bit. This key assumption allows us to derive remarkably explicit expressions for the minimum distribution time for general, heterogeneous models. In a real peer-assisted file distribution system such as BitTorrent, the file is distributed with small chunks, and a node can only forward a chunk once it has fully received the chunk. However, previous work suggests that it is difficult, if not impossible, to obtain closed-form expressions for the minimum distribution time for heterogeneous systems using a chunk-based model. So far, for the chunk-based model, closed form expressions for $T_{\min}$ are only available for very simple cases of nodes having homogeneous bandwidths and infinite download bandwidths [17]. Needless to say, the explicit expressions we derive for $T_{\min}$ with the fluid-based model are lower bounds for more realistic chunk-based models. However, we will show that for homogeneous systems, the percent error in our fluid expressions compared with a chunk-based model is roughly $(\log_2 L)/M$, where M is the number of chunks in the file and L is the number of leechers. We show that this percent error is negligible for situations of practical interest with even medium file sizes. In summary, although our fluid model is somewhat less realistic than the chunk model, it leads to an explicit expression for the minimum distribution time for heterogeneous systems, which closely approximates the minimum distribution time of the chunk-based model for homogeneous systems. We conjecture that it is also a close approximation for heterogeneous systems as well.

As in [5], [21], [17] we make the following assumptions in our model. We assume that the bandwidth bottlenecks are in the access (upload and download rates) and not in the Internet core. This assumption largely holds in today's Internet. Further, we abstract away the effects of network congestion and of TCP congestion control. Thus, our expressions for $T_{\min}$ are actually approximations for real-world file distribution times. The expressions nevertheless provide relevant, back-of-the-envelope calculations that can be used to benchmark a file-distribution protocol for arbitrary upload and download rates.

Again as in [5], [21], [17], [8] we also assume that each node in the system participates in the file distribution up until it has obtained the complete file. This allows us to focus on the key issue of understanding how various system parameters influence the peer-assisted distribution process. We expect that the insights garnered from the static system will contribute to understanding dynamic problems as well.

A. Notation

We introduce the following notation:

- Number of seed nodes $S = |\mathcal{S}|$.
- Number of leecher nodes $L = |\mathcal{L}|$.
- For any subset $\mathcal{A} \subseteq \mathcal{I}$, denote $u(\mathcal{A}) = \sum_{i \in \mathcal{A}} u_i$.
- For any subset $\mathcal{A} \subseteq \mathcal{L}$, denote $d_{\min}(\mathcal{A}) = \min_{i \in \mathcal{A}} d_i$. Let $d_{\min} = d_{\min}(\mathcal{L})$.
- For any subset $\mathcal{L}' \subseteq \mathcal{L}$, denote $T_{\min}(\mathcal{L}')$ as the minimum distribution time for leechers in $\mathcal{L}'$.

III. Minimum Distribution Time

In this section we consider finding $T_{\min}$ for the single-class problem. Before stating the main result, we make some observations. First, because the node with the lowest download rate cannot obtain the file faster than $F/d_{\min}$, we have $T_{\min} \geq F/d_{\min}$. Second, because the set of seeds cannot distribute fresh bits at a rate faster than $u(\mathcal{S})$, a leecher cannot receive the file at a rate faster than $u(\mathcal{S})$, implying $T_{\min} \geq F/u(\mathcal{S})$. Third, because the aggregate upload bandwidth of the system is $u(\mathcal{I})$ and because a total of LF bits needs to be distributed to the leechers, we have $T_{\min} \geq LF/u(\mathcal{I})$. Thus, we have the following lower bound for peer-assisted file distribution:

$$T_{\min} \geq \max\{F/d_{\min}, LF/u(\mathcal{I}), F/u(\mathcal{S})\} \qquad (1)$$

Although the lower bound in (1) is clear, what is not clear at this point is whether this lower bound is tight or loose, and whether there may be other factors – besides the three factors in (1) – that could tighten the lower bound. The following theorem shows that the right-hand side of (1) is not only a lower bound but also the exact value of the minimum distribution time $T_{\min}$. In other words, the following theorem shows that for any upload and download parameters, there is some distribution scheme that actually achieves the right-hand side of (1).

Theorem 1: The minimum distribution time for the general heterogenous peer-assisted file distribution system is

$$T_{\min} = \frac{F}{\min\{d_{\min}, \dfrac{u(\mathcal{I})}{L}, u(\mathcal{S})\}}$$

Observe that this expression is remarkably simple and explicit. It can be used to benchmark the distribution time for any peer assisted file distribution protocol. As an example, consider distributing a 1.25 Gbyte file with 2 seeds each with an upload bandwidth of 10 Mbps. Suppose there is a homogeneous set of leechers with download bandwidth $d = 1$ Mbps and upload rate of u.

u	$L = 10$	$L = 100$	$L = 1000$
200Kbps	2.78 hrs	6.94 hrs	12.63 hrs
400Kbps	2.78 hrs	4.63 hrs	6.61 hrs
600Kbps	2.78 hrs	3.47 hrs	4.48 hrs
800Kbps	2.78 hrs	2.78 hrs	3.39 hrs
1000Kbps	2.78 hrs	2.78 hrs	2.78 hrs

TABLE I

MINIMUM DISTRIBUTION TIME: 1.25GBYTE FILE WITH 2 SEEDS EACH WITH AN UPLOAD BANDWIDTH OF 10 MBPS. L LEECHERS WITH DOWNLOAD BANDWIDTH $d = 1$ MBPS AND UPLOAD BANDWIDTH u.

Using Theorem 1 and making simply calculations by hand, we obtain Table 1, which shows the minimum distribution time for three cases (i) $L = 10$, (ii) $L = 100$ and (iii) $L = 1000$ for several values of u. In Section V we will use Theorem 1 to benchmark the performance for a popular client for BitTorrent.

Proof of Theorem 1: We organize the proof into the following four cases:

- Case A: $d_{\min} \leq \min\{u(\mathcal{I})/L, u(\mathcal{S})\}$ and $d_{\min} \leq u(\mathcal{L})/(L-1)$
- Case B: $d_{\min} \leq \min\{u(\mathcal{I})/L, u(\mathcal{S})\}$ and $d_{\min} \geq u(\mathcal{L})/(L-1)$
- Case C: $u(\mathcal{I})/L \leq \min\{d_{\min}, u(\mathcal{S})\}$
- Case D: $u(\mathcal{S}) \leq \min\{d_{\min}, u(\mathcal{I})/L\}$

Clearly, these 4 cases are exhaustive.

Denote $s_i(t)$ for the rate at which the seeds send bits to leecher i at time t. For each case, we construct a rate profile with the following general structure. As soon as each leecher i begins to receive its bits from the seeds, it replicates the received bits to each of the other $L-1$ leechers at some rate $\leq s_i(t)$ (to be specified for each case) as shown in Figure 1. Thus, for each case, the distribution scheme consists of L appliication-level multicast trees, with each tree rooted in the seeds, passing through one of the leechers, and terminating at each of the $L-1$ other leechers. For each case, we need to show:

1) The seed rate $s_i(t)$ is non-negative and the aggregate seed rate does not exceed the aggregate seed upload bandwidth, that is, $\sum_{i\in\mathcal{L}} s_i(t) \leq u(\mathcal{S})$.
2) The total rate at which a leecher i uploads bits does not exceed its upload bandwidth u_i.
3) The total rate at which a leecher i receives bits does not exceed its download bandwidth d_i.

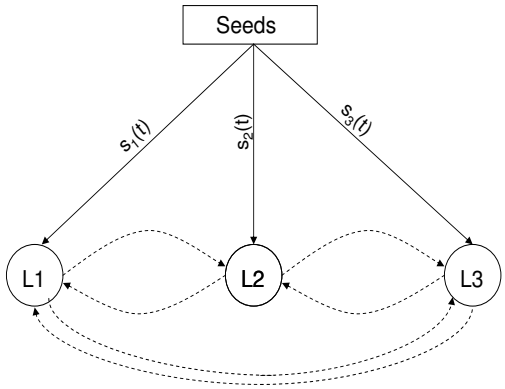

Fig. 1. *General structure of distributing schemes:* Leecher $i \in \{1, 2, 3\}$ downloads fresh data at the rate $s_i(t)$ from the seeds. It then replicates this fresh data at a rate $\leq s_i(t)$ to other 2 leechers.

4) Finally, that the resulting rate profile achieves the lower bound in (1).

In the proofs of the special cases, we will often make use of the following fact: $u(\mathcal{L})/(L-1) \leq u(\mathcal{S})$ if and only if $u(\mathcal{I})/L \leq u(\mathcal{S})$.

Case A: $d_{\min} \leq \min\{u(\mathcal{I})/L, u(\mathcal{S})\}$, $d_{\min} \leq u(\mathcal{L})/(L-1)$

For this case we must show $T_{\min} = F/d_{\min}$. We now construct a rate profile satisfying the four conditions listed above. Have the set of seeds send each of the leechers a different portion of the file with leecher i receiving at rate

$$s_i(t) = (1-\delta)\frac{u_i}{L-1} \text{ for all } t \geq 0$$

where

$$\delta = \{\frac{u(\mathcal{L})}{L-1} - d_{\min}\}/\{\frac{u(\mathcal{L})}{L-1}\}$$

Clearly $\delta \in [0, 1]$ so that $s_i(t) \in [0, u_i/(L-1)]$. Further the above rate profile can be supported by the seeds because

$$\sum_{i\in\mathcal{L}} s_i(t) = d_{\min} \leq u(\mathcal{S}),$$

Now we have each leecher $i \in \mathcal{L}$ replicate $s_i(t)$ to each of the other $L-1$ leechers. This can be done because by doing this leecher i is utilizing only $(1-\delta)u_i$ out of its u_i upload bandwidth available.

Thus, a leecher $i \in \mathcal{L}$ will be downloading fresh content at a total rate of

$$r_i(t) \quad = \quad s_i(t) + \sum_{k\neq i} s_k(t) = d_{\min}$$

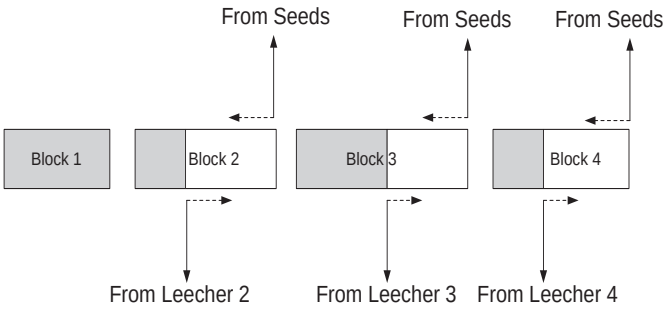

Fig. 2. *Bits at leecher 1 at time $t = W$:* The file is divided into $L = 4$ non-equal-size blocks. Shaded area in each block represents the bits already downloaded by leecher 1 at time $t = W$. Fluid corresponding to the unshaded area arrives at leecher 1 from seeds and other leechers $\{2, 3, 4\}$ for $t \geq W$. Leechers send bits to leecher 1 from beginning of unshaded area; seeds send bits to leecher 1 from end of unshaded area. Both leechers and seeds continue to send bits until leecher 1 has all the bits.

In the above equation the first term is the contribution from seeds and the second term is the contribution from $L - 1$ leechers. This download rate can be maintained at each leecher for all $t \in [0, F/d_{\min}]$. The corresponding distribution time for this rate profile is $F/d_{\min}$. However, according to in-equality in (1) this will imply that the minimum distribution time in this case is simply $T_{\min} = F/d_{\min}$.

Case B: $d_{\min} \leq \min\{u(\mathcal{I})/L, u(\mathcal{S})\}$, $d_{\min} \geq u(\mathcal{L})/(L-1)$

For this case we again must show $T_{\min} = F/d_{\min}$. Have the set of seeds send each of the leechers a different portion of the file with leecher i receiving at rate

$$s_i(t) = \frac{u_i}{L-1} + \delta \text{ for all } t \geq 0$$

where

$$\delta = d_{\min} - \frac{u(\mathcal{L})}{L-1}$$

Clearly $\delta \geq 0$ so that $s_i(t) \geq u_i/(L-1)$. This rate profile can be supported by the seeds because

$$\sum_{i \in \mathcal{L}} s_i(t) = Ld_{\min} - u(\mathcal{L}) \leq u(\mathcal{I}) - u(\mathcal{L}) = u(\mathcal{S})$$

We now have each leecher i transfer at rate $u_i/(L-1)$ to each of the other $L-1$ leechers. Note that this replication strategy does not violate the upload bandwidth constraints of the leechers. Also note, that the forwarding rate $u_i/(L-1)$ at leecher i is less than the rate at which leecher i receives bits from the seeds. Leecher i is thus downloading the file at a total rate of

$$\begin{aligned} r_i(t) &= (\frac{u_i}{L-1} + \delta) + \sum_{k \neq i} \frac{u_k}{L-1} \\ &= \sum_{k \in \mathcal{L}} \frac{u_k}{L-1} + \delta = d_{\min} \end{aligned}$$

With the rate profile shown above it is easy to ensure that each leecher will be downloading distinct bits of the file until

$$W := \frac{F}{\sum_{i \in \mathcal{L}} s_i(t)} = \frac{F}{Ld_{\min} - u(\mathcal{L})}$$

Further at $t = W$ the complete file has been sent by the seeds to the set of the leechers, with each leecher holding distinct portion of the file. Now we show that we can maintain this rate profile while having each leecher continue to download fresh content for $t \geq W$ as well.

Note that each leecher i is sending out fresh data (corresponding to a block of the file) at a rate lower than the rate at which it is receiving. This implies leecher i at time $t = W$ has received all of the corresponding block i of the file but has not completely received the other $L-1$ blocks. This situation is depicted in Figure 2 with $L = 4$ leechers where leecher $i = 1$ has received block 1 (shaded) completely directly from the seeds but has only partially received the other 3 blocks (from leechers {2,3,4}).

For all times $t \geq W$ we require the seeds to send bits from the unshaded areas of the blocks at rate $s_i(t)$ as before. Simultaneously, the other $L-1$ leechers continue to send bits from the unshaded areas of the blocks (which they have already received by time W). For example with the case depicted in Figure 2, the seeds send the unshaded area from blocks 2, 3 and 4 to leecher 1. At the same time the leecher $j \in \{2, 3, 4\}$ sends bits from the unshaded area of block j with $u_j/(L-1)$ rates as before.

In the manner shown above the download rate of $r_i(t) = d_{\min}$ can be maintained at each node for all $t \in [0, F/d_{\min}]$ while having each leecher download fresh data for all these times. The corresponding distribution time for this rate profile is $F/d_{\min}$. However according to in-equality in (1) this will imply the minimum distribution time in this case is simply $T_{\min} = F/d_{\min}$.

Case C: $u(\mathcal{I})/L \leq \min\{d_{\min}, u(\mathcal{S})\}$

Let

$$s_i(t) = \frac{u_i}{L-1} + \delta \text{ for all } t \geq 0$$

where

$$\delta = (u(\mathcal{S}) - \frac{u(\mathcal{L})}{L-1})/L$$

Clearly $\delta \geq 0$. This rate profile can be supported by the seeds because $\sum_{i \in \mathcal{L}} s_i(t) = u(\mathcal{S})$.

We have each leecher $i \in \mathcal{L}$ replicate $u_i/(L-1)$ to the other $L-1$ leechers without violating upload bandwidth constraints. Thus, a leecher i will be downloading fresh content at a total rate of

$$\begin{aligned} r_i(t) &= (\frac{u_i}{L-1} + \delta) + \sum_{k \neq i} \frac{u_k}{L-1} \\ &= \sum_{k \in \mathcal{L}} \frac{u_k}{L-1} + \delta = \frac{u(\mathcal{I})}{L} \leq d_{\min} \end{aligned}$$

Thus, this rate can be supported by the leecher's download bandwidth. Let $W = F/u(\mathcal{S})$. Notice that as in Case B, at $t = W$ the complete file has been sent by the seeds to the set of the leechers $\mathcal{L}$ with each leecher holding a distinct portion of the file. With a scheduling scheme similar to case B and also depicted in Figure 2, we can ensure that for all times $t \geq W$ all leechers are downloading bits they have not downloaded before while maintaining the rate profile constructed above. Furthermore, since each node is downloading at an equal rate of $u(\mathcal{I})/L$, we have $\sum_{i \in \mathcal{L}} r_i(t) = u(\mathcal{I})$ for all $t \in [0, LF/u(\mathcal{I})]$. The corresponding distribution time is given by $LF/u(\mathcal{I})$. However, according to in-equality in (1) this implies that the minimum distribution time is simply $T_{\min} = LF/u(\mathcal{I})$.

Case D: $u(\mathcal{S}) \leq \min\{d_{\min}, u(\mathcal{I})/L\}$

Let

$$s_i(t) = (1-\delta)\frac{u_i}{L-1} \text{ for all } i \in \mathcal{L}$$

where

$$\delta = \{\frac{u(\mathcal{L})}{L-1} - u(\mathcal{S})\}/\frac{u(\mathcal{L})}{L-1}$$

Clearly $\delta \in [0,1]$. Because $(L-1)s_i(t) \leq u_i$ for all $i \in \mathcal{L}$ each leecher i can transfer $s_i(t)$ to the other $L-1$ leechers. Further this rate profile can be supported by the seeds because

$$\sum_{i \in \mathcal{L}} s_i(t) = u(\mathcal{S}) \tag{2}$$

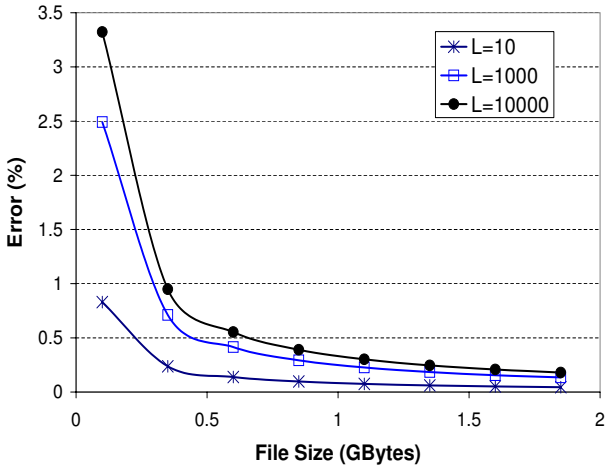

Fig. 3. *Single Class:* Percentage error between fluid-based and chunk-based model. For typical file sizes ($\geq$ 350 MB) the error is less than 1 percent

Given that, the download rate of fresh content for leecher i is

$$r_i(t) = s_i(t) + \sum_{k \neq i} s_k(t) = u(\mathcal{S}) \leq d_i$$

Since each leecher is downloading at an equal rate this download rate can be maintained at each leecher for all $t \in [0, F/u(\mathcal{S})]$. The corresponding distribution time is given by $F/u(\mathcal{S})$. However, according to in-equality in (1) this implies that the minimum distribution time is simply $T_{\min} = F/u(\mathcal{S})$.□

It should be noted that, for each of the four cases, the distribution scheme given in the proof of Theorem 1 is such that all leechers complete their downloads at the same time.

IV. Insights for the Single Class Problem

In this section we examine the implications of Theorem 1.

A. Comparison to Peer-Assisted File Distribution with Chunks

In a real peer-assisted file distribution system such as BitTorrent, the file is divided into chunks, and a node can only forward a chunk once it has fully received the chunk. A natural question then is whether the fluid model (as studied in this paper) grossly underestimates the minimum distribution time for when chunks are used? We now present the percent error in $T_{\min}$ due to lack of store-and-forwarding of chunks in the fluid based model. For comparison we choose a homogeneous system with peers having infinite download capacity. The reason for

this choice is that closed form expressions for chunk based models are available for this simplistic situation [17]. (No such expression is available for heterogeneous systems with chunks.)

Suppose the file consists of M equal size chunks. Further suppose $S = 1$ and all the upload capacities are homogeneous ($u_i = u$ for all $i \in \mathcal{I}$) and all the download capacities are infinite($d_i = \infty$ for all $i \in \mathcal{L}$). For this homogeneous model, the minimum distribution time taking chunks into account is given by [17], [3]:

$$T_{\min}\{\text{Chunk}\} = \frac{F}{u}\left(1 + \frac{\log_2 L}{M}\right) \tag{3}$$

For our fluid model, for this special case Theorem 1 becomes

$$T_{\min}\{\text{Fluid}\} = \frac{F}{u} \tag{4}$$

From equations (3) and (4), we obtain the fractional error between chunk based and fluid based model:

$$\frac{T_{\min}\{\text{Chunk}\} - T_{\min}\{\text{Fluid}\}}{T_{\min}\{\text{Fluid}\}} = \frac{\log_2 L}{M} \tag{5}$$

Let us now consider the fractional error for some realistic scenarios. The typical chunk size in BitTorrent is 256 KBytes [15]. Figure 3 shows the percent error for different values of L and different file sizes. From the figure we observe that for file size of 350 MBytes or more, the percentage error between $T_{\min}\{\text{Fluid}\}$ and $T_{\min}\{\text{Chunk}\}$ is less than 1%, even when there are 10,000 leechers.

From (5) it follows that, for homogeneous systems, the error can be safely ignored if $M >> \log_2 L$. For typical file sizes of the order of several Gbytes, this condition is easily satisfied. We conjecture this is true for heterogeneous systems as well. Hence, even though the fluid model is not as realistic as the chunk model, it is highly accurate and has the advantage of providing a simple, explicit expression for the minimum distribution time for general, heterogeneous systems. Further, if every leecher has one distinct chunk available then it is easy to see that the rest of the chunks can be distributed using the distribution scheme from Theorem 1 as if it was a fluid transfer.

B. Advantages over Client-Server model

For a client-server based file distribution, the minimum distribution time is

$$T^{cs}_{\min} = \frac{F}{\min\{d_{\min}, u(\mathcal{S})/L\}}$$

For $d_{\min}$ large and $u_i = u$, for all $i \in \mathcal{L}$, we can compare distribution times for both client-server and peer-assisted based file distribution. In this case, $T_{\min}$ for peer-assisted distribution and client-server is respectively,

$$T^{pa}_{\min} = \frac{F}{\min\{\frac{Lu + u(\mathcal{S})}{L}, u(\mathcal{S})\}}$$

$$T^{cs}_{\min} = \frac{F}{u(\mathcal{S})/L}$$

This plot is shown in Figure 4. For this plot we choose

$$\frac{F}{u} = 10\frac{F}{u(\mathcal{S})} = 1$$

and plot the corresponding distribution times for for varying value of L.

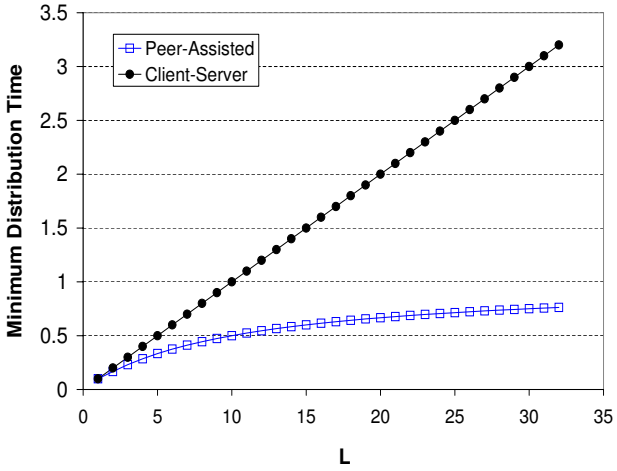

Fig. 4. *Single Class:* Comparison of minimum distribution time for client-server and peer-assisted file distribution.

C. Departing Leechers

Consider a variation of the minimum distribution time problem in which leechers depart some time after they receive the entire file. Specifically, suppose leecher i leaves τ_i seconds after receiving entire file, where τ_i is a random variable with distribution $F_i(\tau)$. A natural problem is to minimize the expected distribution time. In the distribution scheme in the proof of Theorem 1, all leechers finish at the same time. It follows that even when leechers randomly depart after receiving the file, our distribution scheme is optimal and the minimum distribution time is still given by Theorem 1.

D. Fairness Issues

We wish to characterize the contribution of a leecher i in terms of the rate at which it replicates fresh data to all other leechers. One can observe that in all the rate profiles ($r_i(t)$ for all $i \in \mathcal{L}$) used in the proof of

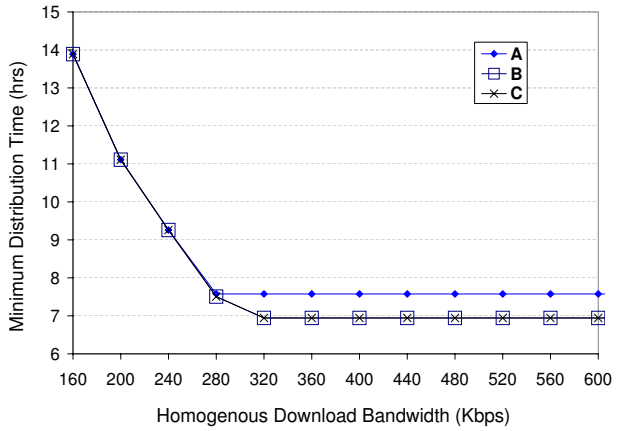

Fig. 5. *Single Class:* Three different sets of leechers $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ with common homogenous download bandwidth d are chosen. The minimum distribution time is plotted while d is varied on the x-axis. See Section IV-E

Theorem 1, this contribution either takes the form of $Ks_i(t)$ or of u_i where $K \in (0, 1)$ is some constant and is the same for every leecher. Roughly speaking, at best the leechers are contributing up to what is allowed by their upload capacity or at worst are contributing an equal percentage of their respective receiving rates of fresh data in order that the rate profile achieves minimum distribution time. Thus one can conclude that the goals of achieving minimum distribution time and ensuring fairness of bandwidth contribution by the leechers are not necessarily contradictory.

E. Influence of Download Bandwidth on Minimum Distribution time

If the minimum download bandwidth is small, then it heavily influences the value of minimum distribution time. To understand this issue better we present the following numerical example. We select the following values for the file size and aggregate seed bandwidth, $F = 1$ Gbyte, $u(\mathcal{S}) = 320$ Kbps. We choose three different set of leechers $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$. The leechers in these sets have the following upload rates, (i) $\mathcal{A} = \{80, 160, 320\}$ kbps, (ii) $\mathcal{B}= \{80, 160, 480\}$ kbps and (iii) $\mathcal{C}= \{80, 160, 600\}$ kbps. In Figure 5 we vary the homogenous download bandwidth d along the x-axis and plot the minimum distribution time on the y-axis. Some observations:

- Note that for set A of leechers, we have $u(\mathcal{I})/L < u(\mathcal{S})$. For lower values of d the value of $T_{\min}$ decreases as in F/d. Once $d = u(\mathcal{I})/L = 36.667$ is reached, $T_{\min} = LF/u(\mathcal{I}) = 27.27$ remains constant.
- Set B and C of leechers have exactly the same minimum distribution time for all values of d. This is to be expected since for both sets of leechers, $u(\mathcal{I})/L \geq u(\mathcal{S})$ and from Theorem 1, in this case the minimum distribution time is not a function of $u_i, i \in \mathcal{L}$ at all. As expected increasing d beyond $u(\mathcal{S}) = 40$ does not decrease the minimum distribution time any further.

V. Benchmarking BitTorrent Clients

Having derived the the optimal distribution time for peer-assisted content distribution, we now apply the theory to a real P2P content-distribution scheme. Specially, we now examine how the BitTorrent client Azureus in practice stacks up against the optimal distribution time given in Theorem 1. To this end, we use Azureus to distribute a file from 1 seed host to 10 leecher hosts, measure the distribution time, and compare the distribution time to the optimal distribution time. A comprehensive measurement study of BitTorrent is beyond the scope of this paper. Instead, the aim of this measurement study is to show how Theorem 1 can be used to benchmark a real distribution scheme.

A. Test-Bed Setup

Our test-bed consists of 11 PCs, each running the BitTorrent client Azureus version 2.3.0.6 [2] on top of Fedora Linux. One of the PCs is a seed, initially having the entire file. The other PCs are leechers, initially not having any portion of the file. The PCs are interconnected by a multiport Ethernet switch, with 100Mbps access per port. Thus the test environment is closed, with no background traffic. We chose Azureus as the BitTorrent client because (a) it allows rate throttling capability for both uploading and downloading, (b) it is an open source software which permitted us to make some minor modifications, (c) it is very popular with over 5 million downloads [12] and has been crowned the most popular open-source software by SourceForge.net [13]. We first show in the next subsection that Azureus's bandwidth throttling capability is sufficiently accurate for our measurement purposes.

B. Accuracy of Throttling

Azureus does rate throttling at the application layer. It does this by implementing a simple token bucket control algorithm on top of receiving/sending TCP sockets. When a token is not available for uploading, it delays socket writes until a token becomes available. Similarly, when a token is not available for downloading, it delays socket reads until a token becomes available.

To measure the accuracy of Azureus's throttling, we did experiments on both download and upload rate throttling. The experiments consisted of one seed and one leecher only. To evaluate download rate throttling, we constrained the download rate at the leecher without constraining the seed's upload rate. Similarly, to evaluate upload rate throttling, we constrained upload rate at the seed without constraining the download rate of the leecher. For these experiments, we selected two different rates, 25 KB/s and 45 KB/s,giving four different scenarios. For each scenario, we used a file size of $F = 62.765$ MB, noted the time taken to complete the transfers, and calculated the observed rate as

$$\text{observed rate} = \frac{F}{\text{completion time}}$$

From this, we calculated the error in bandwidth throttling as

$$\text{throttle error} = \frac{\text{observed rate - throttle rate}}{\text{throttle rate}}$$

For each scenario, we ran the experiment 6 times. In table II, we provide the 6 throttle errors for each of the 4 scenarios. We see that the throttling errors are small. Thus, we can safely approximate Azureus's specified throttling rates as the actual maximum upload and download rates.

C. Benchmarking Experiments

Having verified Azureus's throttling mechanism, we now describe the results of the main benchmarking experiments. We experimented with three different rate profiles.

- *Profile 1 :* $u_i = d_i = 45$ KB/s for all leechers, $u(\mathcal{S}) = 80$ KB/s. Here the minimun download rate, $d_{\min}$, is the bottleneck.
- *Profile 2 :* $d_i = 45$ KB/s, $u_i = 25$ KB/s for all leechers, $u(\mathcal{S}) = 80$ KB/s. Here normalized total upload rate, $u(\mathcal{I})/L$, is the bottleneck.
- *Profile 3:* $u_i = d_i = 45$ KB/s for all leechers, $u(\mathcal{S}) = 35$ KB/s. Here the seeds' upload rate, $u(\mathcal{S})$, is the bottleneck.

For each experiment, we again used a file size of $F = 62.765$ MB. We measured the observed minimum distribution time ($T_{\min}^{(obs)}$) and also calculate the theoretical minimum distribution time ($T_{\min}$) using Theorem 1. Since $T_{\min}$ is optimal, it serves as a lower bound for the observed value. We use percentage overshoot as a metric to measure the deviation between the theoretical and observed distribution times.

$$\text{Overshoot (\%)} = \frac{T_{\min}^{(obs)} - T_{\min}}{T_{\min}} \times 100$$

Throttle Rate	Download (%)	Upload(%)
45KB/s	4.09, -0.73, -0.37	2.41 2.41 2.78
	3.70, -1.08, -1.01	2.63, 2.33 2.18
25KB/s	2.06, 1.89, 1.19,	0.67, 0.71, 0.63,
	-0.45, -2.54, 1.36,	0.71, 0.67, 0.71

TABLE II
Throttling Accuracy: Throttle Error Samples

For a given profile, the randomness in the BitTorrent algorithm generates a different observed distribution time for each repeated experiment. Thus, the overshoot can be viewed as a random variable with some mean. To estimate this mean, for each profile, we performed 12 measurements, generating 12 sample $T_{\min}^{(obs)}$. From the 12 samples, we constructed confidence intervals for the true mean overshoot.

Azureus allows the user to set the maximum number of simultaneous unchoked upload connections from any leecher. The default setting for this variable is 4. One would expect that increasing this setting would decrease the minimum distribution time. This indeed turns out to be the case. We did measurements for each of the bandwidth profiles with both 4 and 10 unchoked upload connections (denoted as m).

The corresponding confidence intervals for the percent overshoot are shown in Figure 6. We make the following observations:

- For this modest-size network, BitTorrent is fairly efficient when either $d_{\min}$ or $u(\mathcal{I})/L$ is the bottleneck, with the actual distribution time typically not more than 30% of the theoretical optimal distribution time. When $u(\mathcal{S})$ is the bottleneck, however, BitTorrent is less efficient, with distribution times often exceeding 60% of the theoretical optimal.
- The overshoot generally decreases when we increase the number of unchoked upload connections.

The overshoot is the highest when the seed bandwidth is the bottleneck (profile 3). From our measurements there seems to be evidence that BitTorrent is not very efficient at distributing data when the source bandwidth itself is in short supply. We observed that the throughput for each leecher was good in the middle of the file transfer and poor in the beginning and towards the end. It seems that the leechers are not able to download from each other either because the unchoked neighbors have too many overlapping sets of chunks or they have choked

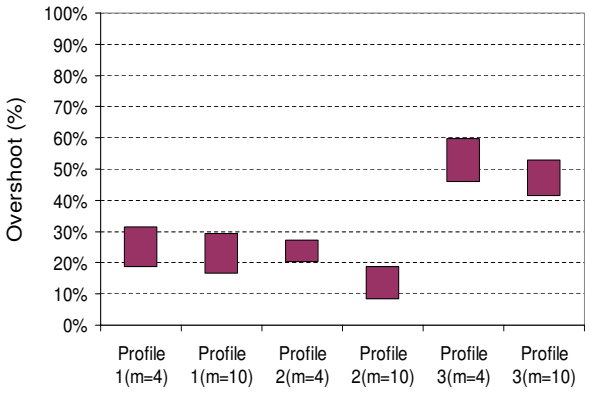

Fig. 6. *10 Leechers: 95% Confidence Intervals corresponding to percent Overshoot*

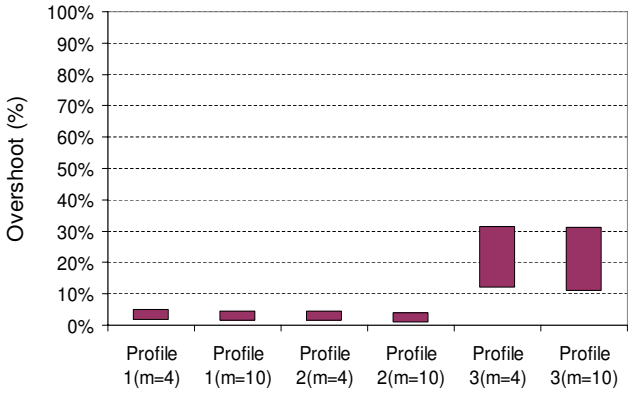

Fig. 7. *5 Leechers: 95% Confidence Intervals corresponding to percent Overshoot*

leechers that may have interesting chunks to offer. Hence in the beginning of the transfer, the peer-assisted file distribution process more or less resembles the client-server distribution. However, once the leechers collect enough diversity of chunks they can collaborate more effectively and the throughput at each leecher increases. Towards the end of the transfer we see the familiar situation where each leecher is missing the overlapping small sets of chunks and again the distribution process degrades to that in client-server model.

The seed bandwidth is not the bottleneck in the other two profiles. Hence, when leechers have overlapping sets of chunks, the seed bandwidth is sufficient to ensure that the distribution process does not degrade significantly.

With the typical bandwidths available to end-user systems, one would expect that average upload leecher bandwidths $(u(\mathcal{I})/L)$ to be the bottleneck factor. Thus, we believe that profile 2 is the most common situation encountered in real-world file-transfers. One can observe that BitTorrent performs admirably well in this scenario for the modest-size test network. We also show the corresponding statistics for a test-bed of 5 leechers in Figure 7

VI. Related Work

We now review the related work for single-class peer-assisted file distribution. In what follows, we discuss modeling (rather than "systems") papers relating to peer-assisted file distribution.

Biersack, Rodriguez and Felber obtain the peer-assisted distribution time for three overlay topologies (linear, tree and forest) for the chunk-based model [5]. They make the simplifying assumption that all peers (seeds and leechers) have homogeneous and equal upload and download bandwidths. Cherkasova and Lee propose FastReplica, an algorithm for distributing a large file from a source node to receiving nodes, using the assistance of the upload capacity of the receiving nodes [8]. In this algorithm the file is divided into equal parts and transferred to each node, after which the nodes share the file among each other. The paper also does performance analysis assuming specific bandwidth profiles among the receiving nodes. Neither [5] nor [8] consider the problem of finding the minimum distribution time.

To our knowledge, Mundinger and Weber are the only others to attempt to obtain the minimum distribution time for peer-assisted file distribution [17]. They use both chunk-based and fluid-based model. For their fluid model, they succeed at determining the minimum distribution time for the case in which all all download bandwidths are infinite. They also give some insights for the case with finite download bandwidths under the assumption of homogeneous upload bandwidths.

There are also some additional modeling papers for peer-assisted file distribution that do not address file distribution times. Yang and Veciana study service capacity of a P2P network [21]. They model the propagation of the file as a branching process. The paper evaluates average delay experienced by peers in a transient and a steady state regime. The paper shows that in the steady state, the service capacity of a P2P system will scale with and track the offered loads. Qui and Srikant develop an entirely different type of fluid model for BitTorrent-like systems [18]. In their model, the number of seeds and leechers is dynamic, and the fluid is not bits but the number seeds and leechers participating. They develop a differential equation for the fluid model, from which they determine the performance of the dynamic system. Massoulié and Vojnović develop a probabilistic coupon replication model for a BitTorrent-like system [16]. The

coupons can be thought of as chunks. Each user is characterized by its current coupon collection, and the system evolution is then specified by how users of distinct types meet and, which coupons get replicated upon such encounters. In asymptotic regimes, they consider peering strategies for minimizing the mean file download time. Bharambe et al. [4] simulate BitTorrent with thousands of nodes and analyze its performance.

In [14], we have extended the current paper to cover peer-assisted file distribution with differentiated services. Specifically, we suppose there are two classes of receiving nodes, with the goal being to get the file as quickly as possible to the first-class nodes. We again derive explicit expressions for the distribution time, and argue that differentiated service quality can be provisioned at the application level.

VII. Conclusion

Peer-assisted file distribution is an important application in the Internet today. A fundamental problem in peer-assisted file distribution is to determine the minimum achievable time to distribute a file to all receiving nodes. For a model in which discrete chunks are stored and forwarded at the nodes, this fundamental problem becomes a complex optimal scheduling problem.

In this paper we first consider a fluid-based version for the problem of determining the minimum achievable distribution time. With this fluid model, we obtain an explicit expression for this fundamental problem. Being simple and insightful, the result could be taught in an introductory course on computer networking or distributed systems. We argue that little accuracy is lost in using a fluid-based model instead of the more realistic chunk-based model.

These results open a number of avenues for future research. One directions is compare the fluid minimum distribution time with the chunk-based distribution time for heterogenous systems. Another direction is to determine the minimum distribution time for systems which limit the number of simultaneous connections between participating nodes.

VIII. Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. 0325777. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] Avalanche: File Swarming with Network Coding, " *http://research.microsoft.com/ pablo/avalanche.htm*"

[2] http://www.azureus.com

[3] A. Bar-Noy, S. Kipnis, and B. Schieber, "*Optimal multiple message broadcasting in telephone-like communication systems,*" Discrete Applied Mathematics, 100:1-15,2000.

[4] A.R. Bharambe, C. Herley, and V.N.Padmanabhan, "*Analyzing and Improving a BitTorrent Network's Performance Mechanisms,*" IEEE INFOCOM 2006.

[5] E.W. Biersack, P. Rodriguez and P. Felber, "*Performance Analysis of Peer-to-Peer Networks for File Distribution,*" In proceedings of Quality of Future Internet Services (QOFIS04), September 2004.

[6] Cachelogic, "*http://www.cachelogic.com/research*"

[7] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "*SplitStream: High-bandwidth multicast in a cooperative environment,*" ACM Symposium on Operating Systems Principles (SOSP), October 2003.

[8] L. Cherkasova and J. Lee, "*FastReplica: Efficient Large File Distribution within Content Delivery Networks,*" 4th USENIX Symposium on Internet Technologies and Systems, March 2003.

[9] B. Cohen, "*Incentives Build Robustness in BitTorrent,*" First Workshop on Economics of Peer-to-Peer Systems, May 2003.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, " *Introduction to Algorithms,*" Second Edition, MIT Press and McGraw-Hill, 2001.

[11] C. Gkantsidis, P. Rodriguez, "*Network Coding for Large Scale Content Distribution,*" IEEE INFOCOM 2005, March 2005.

[12] http://www.download-it-free.com/azureus/

[13] http://www.itnews.com.au/newsstory.aspx?CIaNID=31480

[14] R. Kumar, and K.W. Ross, *"Optimal Peer-Assisted File Distribution: Single and Multi-class Problems,"* Submitted.

[15] A. Legout, G. Urvoy-Keller, and P. Michiardi, "*Understanding BitTorrent: An Experimental Perspective,*" Technical Report, INRIA, Sophia Antipolis, November 2005.

[16] L. Massoulié and M. Vojnović, "*Coupon Replication Systems,*" ACM SIGMETRICS 2005.

[17] J. Mundinger, R. R. Weber and G. Weiss, "*Analysis of Peer-to-Peer File Dissemination,*" To appear in Performance Evaluation Review, Eighth Workshop on Mathematical Performance Modeling and Analysis (MAMA 2006) Issue.

[18] D. Qiu and R. Srikant, "*Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks,*" ACM SIGCOMM, September 2004.

[19] R. Sherwood, R. Braud and B. Bhattacharjee, "*Slurpie: A cooperative bulk data transfer protocol,*" IEEE INFOCOM 2004.

[20] Swarmcast, "*http://swarmcast.net*

[21] X. Yang and G. Veciana, "*Service Capacity of Peer-to-Peer Networks,*" IEEE INFOCOM 2004.

[22] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, "*The Bittorrent P2P File-Sharing System: Measurements And Analysis ,*" 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), Feb 2005.

[23] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garces-Erice, "*Dissecting BitTorrent: Five Months in a Torrent's Lifetime,*" Passive and Active Measurements, Antibes Juan-les-Pins, France, April 2004.