Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks

Dongyu Qiu and R. Srikant
Coordinated Science Lab
and
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{dqiu,rsrikant}@uiuc.edu

Abstract

In this paper, we develop simple models to study the performance of BitTorrent, a second generation peer-to-peer (P2P) application. We first present a simple fluid model and study the scalability, performance and efficiency of such a file-sharing mechanism. We then consider the built-in incentive mechanism of BitTorrent and study its effect on network performance. We also provide numerical results based on both simulations and real traces obtained from the Internet.

1 Introduction

Peer-to-Peer (P2P) applications have become immensely popular in the Internet. Traffic measurements shows that P2P traffic is starting to dominate the bandwidth in certain segments of the Internet [2]. Among P2P applications, file sharing is perhaps the most popular application. Compared to traditional client/sever file sharing (such as FTP, WWW), P2P file sharing has one big advantage, namely, scalability. The performance of traditional file sharing applications deteriorates rapidly as the number of clients increases, while in a well-designed P2P file sharing system, more peers generally means better performance. There are many P2P file sharing programs, such as Kazza, Gnuttella, eDonkey/overnet, BitTorrent, to name a few. In this paper, we develop simple models to understand and study the behavior of BitTorrent [8] which is proving to be one of the more popular P2P applications today.

For a BitTorrent network (or a general P2P file sharing network), several issues have to be addressed in order to understand the behavior of the system.

- *Peer Evolution:* In P2P file sharing, the number of peers in the system is an important factor in determining network performance. Therefore, it is useful to study how the number of peers evolves as a function of the request arrival rate, the peer departure rate, the uploading/downloading bandwidth of each peer, etc.

- *Scalability:* To realize the advantages of P2P file sharing, it is important for the network performance to not deteriorate, and preferably to actually improve, as the size of the network increases. Network performance can be measured by the average file downloading time and the size of the network can be characterized by the number of peers, the arrival rate of peers, etc.

- *File Sharing Efficiency:* It is common for peers in a P2P network to have different uploading/downloading bandwidths. Further, in BitTorrent-like systems, a file may be broken into smaller pieces and the pieces may be distributed at random among the peers in the network. To efficiently download the file, it is important to design the file-sharing protocol such that each peer is matched with others who have the pieces of the file that it needs and further, to ensure that the downloading bandwidth of each peer is fully utilized.

- *Incentives to prevent free-riding:* Free-riding is a major cause for concern in P2P networks. Free-riders are peers who try to download from others while not contributing to the network, i.e., by not uploading to others. Thus, most P2P networks try to build in some incentives to deter peers from free-riding. Once the incentive mechanism is introduced into the network, each peer may try to maximize its own net benefit within the constraints of the incentive mechanism. Thus, it is important to study the effect of such behavior on the network performance.

1.1 Relationship to prior work

The basic idea of P2P network is to have peers participate in an application level overlay network and operate as both servers and clients. Since the service burden is distributed to all participating peers, the system is expected to scale well even when the network is very large. Besides file sharing, P2P overlays have also been deployed in distributed directory service [18, 21], web cache [15], storage [9], and grid computation [1] etc.

While early work on P2P systems has mainly focused on system design and traffic measurement [19, 20, 17], some recent research has emphasized performance analysis. In [13], a closed queueing system is used to model a general P2P file sharing system and basic insights on the stationary performance are provided. In [6, 7], a stochastic fluid model is used to study the performance of P2P web cache (SQUIRREL) and cache clusters. A part of our work is motivated by the models in [11, 24], where a branching process is used to study the service capacity of BitTorrent-like P2P file sharing in the transient regime and a simple Markovian model is presented to study the steady-state properties. Our work differs from [11, 24] in the following respects:

- Instead of studying the Markov chain numerically, we develop a simple deterministic model which allows us to obtain simple expressions for the average file-transfer time, thus providing insight into the performance of the P2P network. We also incorporate realistic scenarios in our fluid model such as the abandonment of file transfers by peers and download bandwidth constraints.
- Then, we develop a simple stochastic fluid model which characterizes the variability of the number of peer around the equilibrium values predicted by the deterministic fluid model.
- We also develop a simple model to study the efficiency of downloading from other peers and argue that the file-sharing protocol in BiTorrent is very efficient.
- Finally, we consider the mechanisms built into BitTorrent to avoid free-riding and study the impact of these mechanisms on the users' behaviors and network performance.

2 A brief description of BitTorrent

BitTorrent is a P2P application whose goal is to facilitate fast downloads of popular files. Here we provide a brief description of how BitTorrent operates when a single file is downloaded by many users. Typically the number of simultaneous downloaders for popular files could be of the order of a few hundreds while the total number of downloaders during the lifetime of a file could be of the order of several tens or sometimes even hundreds of thousands. The basic idea in BitTorrent is to divide a single large file (typically a few 100 MBytes long) into pieces of size 256 KB each. The set of peers attempting to download the file do so by connecting to several other peers simultaneously and download different pieces of the file from different peers.

To facilitate this process, BitTorrent uses a centralized software called the *tracker*. In a BitTorrent network, a peer that wants to download a file first connects to the tracker of the file. The tracker then returns a random list of peers that have the file. The downloader then establishes a connection to these other peers and finds out what pieces reside in each of the other peers. A downloader then requests pieces which it does not have from all the peers to which it is connected. But each peer is allowed to upload only to a fixed number (default is four) at a given time. Uploading is called *unchoking* in BitTorrent. Which peers to unchoke is determined by the current downloading rate from these peers, i.e., each peer uploads to the four peers that provide it with the best downloading rate even though it may have received requests from more than four downloaders. This mechanism is intended to deter free-riding. Since a peer is only uploading four other peers at any time, it is possible that a peer, say Peer A, may not be uploading to a peer, say Peer B, which could provide a higher downloading rate than any of the peers to which Peer A is currently uploading. Therefore, to allow each peer to explore the downloading rates of other peers, BitTorrent uses a process called *optimistic unchoking*. Under optimistic unchoking, each peer randomly selects a fifth peer from which it has received a downloading request and uploads to this peer. Thus, including optimist unchoking, a peer may be uploading to five other peers at any time. Optimistic unchoking is attempted once every 30 seconds and to allow optimistic unchoking while keeping the maximum number of uploads equal to five, an upload to the peer with the least downloading rate is dropped.

BitTorrent distinguishes between two types of peers, namely *downloaders* and *seeds*. Downloaders are peers who only have a part (or none) of the file while seeds are peers who have all the pieces of the file but stay in the system to allow other peers to download from them. Thus, seeds only perform uploading while downloaders download pieces that they do not have and upload pieces that they have. Ideally, one would like an incentive mechanism to encourage seeds to stay in the system. However, BitTorrent currently does not have such a feature. We simply analyze the performance of BitTorrent as is.

In practice, a BitTorrent network is a very complicated system. There may be hundreds of peers in the system. Each peer may have different parts of the file. Each peer may also have different uploading/downloading bandwidth. Further, each peer only has partial information of the whole network and can only make decisions based on local information. In addition, BitTorrent has a protocol (called the *rarest-first policy*) to ensure a uniform distribution of pieces among the peers and protocols (call the *endgame mode*) to prevent users who have all but a few of the pieces from waiting too long to finish their download. As with any good modelling exercise, we tradeoff between the simplicity of the model and its ability to capture all facets of the protocol. Thus, we will first use a simple fluid model to study the scalability and the stability of the system. We will then assume that each peer has the global information and study the incentive mechanism of BitTorrent. We will finally briefly study the effect of the optimistic unchoking on free-riding.

3 A Simple Fluid Model

Our model for file-sharing is influenced by the model in [11]. However, while [11] only uses the model to develop a Markov chain which is then studied numerically, we use the key modelling idea in [11] to develop a simple deterministic fluid model which is amenable to analysis and provides insights into the system performance.

In our model, we use the following quantities to capture a BitTorrent peer-to-peer network [8] that serves a given file (without loss of generality, we assume that the file size is 1):

$x(t)$ number of downloaders (also known as leechers) in the system at time t.

$y(t)$ number of seeds in the system at time t.

λ the arrival rate of new requests.

μ the uploading bandwidth of a given peer. We assume that all peers have the same uploading bandwidth.

c the downloading bandwidth of a given peer. We assume that all peers have the same downloading bandwidth and $c \geq \mu$.

θ the rate at which downloaders abort the download.

γ the rate at which seeds leave the system.

η indicates the effectiveness of the file sharing, which we will describe shortly. η takes values in $[0, 1]$.

In a BitTorrent-like P2P network, a downloader can upload data to other peers even though it may only have parts of a file. The parameter η is used to indicate the effectiveness of this file sharing. If there is no constraint on downloading bandwidth, the total uploading rate of the system can be expressed as $\mu(\eta x(t) + y(t))$. If $\eta = 0$, then the downloaders do not upload data to each other and only download from seeds. When the downloading bandwidth constraint is considered, the total uploading rate will be $\min\{cx(t), \mu(\eta x(t) + y(t))\}$.

Next, we comment on the parameters θ and γ. A downloader may not stay in the system till it completely downloads the file. Occasionally, a downloader may leave the network before the downloading is complete if he/she feels that the download is taking too long. We use the parameter $1/\theta$ to denote the average amount of time that a downloader will stay in the system before deciding to abort the download. Equivalently, θ is the rate at which downloaders abort their download and leave the system. In a fluid model, the rate of departures of downloaders will be given be

$$\min\{cx(t), \mu(\eta x(t) + y(t))\} + \theta x(t).$$

While the departures that occur due to the fact that the file download has been completed will become seeds instantaneously, the remaining downloaders will permanently leave the system. The parameter γ is the rate at which seeds depart from the network. Clearly, γ will have an effect on system performance: the lower the γ, the lower the download times since this means that there will more seeds in the system. This parameter γ can be influenced by providing incentives for users to stay in the system after they have downloaded the file, i.e., after they have become seeds. However, BitTorrent currently does not have such incentives and therefore, we simply consider γ to be a fixed constant.

Now, we are ready to describe the evolution of x and y based on the above model. A deterministic fluid model for the evolution of the number of peers (downloaders and seeds) is given by

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \lambda - \theta x(t) - \min\{cx(t), \mu(\eta x(t) + y(t))\},\\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \min\{cx(t), \mu(\eta x(t) + y(t))\} - \gamma y(t), \qquad (1)\end{aligned}$$

along with the obvious constraint that $x(t)$ and $y(t)$ should be non-negative. A key contribution of [11] was to describe the efficiency of data transfer from other downloaders using the parameter η. Our fluid model provides a simple description of the system that was described by a Markov chain in [11]. In addition, we have incorporated other realistic scenarios such as departures of downloaders due to impatience with the downloading process (described by θ) and downloading bandwidth constraint c. In a later subsection, we will also present a simple stochastic fluid model that characterizes the variability around the fluid model. We now study the steady-state performance of the P2P system using the above fluid model.

3.1 Steady-State Performance

To study the system in steady-state, we let

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\mathrm{d}y(t)}{\mathrm{d}t} = 0$$

in (1) and obtain

$$\begin{aligned}0 &= \lambda - \theta\bar{x} - \min\{c\bar{x}, \mu(\eta\bar{x} + \bar{y})\},\\ 0 &= \min\{c\bar{x}, \mu(\eta\bar{x} + \bar{y})\} - \gamma y(t), \qquad (2)\end{aligned}$$

where $\bar{x}$ and $\bar{y}$ are the equilibrium values of $x(t)$ and $y(t)$ respectively.

We first assume $\eta > 0$. Further, suppose that the downloading speed is the constraint, i.e., $c\bar{x} \leq \mu(\eta\bar{x}+\bar{y})$. Equation (2) then becomes a simple linear equation. Solving the equation, we have

$$\begin{aligned}\bar{x} &= \frac{\lambda}{c(1+\frac{\theta}{c})}\\ \bar{y} &= \frac{\lambda}{\gamma(1+\frac{\theta}{c})}. \qquad (3)\end{aligned}$$

Now, the assumption that $c\bar{x} \leq \mu(\eta\bar{x}+\bar{y})$ is equivalent to

$$\frac{1}{c} \geq \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma}).$$

Instead, if we assume that the uploading bandwidth is the constraint, i.e., $c\bar{x} \geq \mu(\eta\bar{x}+\bar{y})$, we get

$$\begin{aligned}\bar{x} &= \frac{\lambda}{\nu(1+\frac{\theta}{\nu})}\\ \bar{y} &= \frac{\lambda}{\gamma(1+\frac{\theta}{\nu})}, \qquad (4)\end{aligned}$$

where $\frac{1}{\nu} = \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$. From $c\bar{x} \geq \mu(\eta\bar{x}+\bar{y})$, we have

$$\frac{1}{c} \leq \frac{1}{\nu} = \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma}).$$

Define $\frac{1}{\beta} = \max\{\frac{1}{c}, \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})\}$, then (3) and (4) can be combined to yield

$$\begin{aligned}\bar{x} &= \frac{\lambda}{\beta(1+\frac{\theta}{\beta})}\\ \bar{y} &= \frac{\lambda}{\gamma(1+\frac{\theta}{\beta})}. \qquad (5)\end{aligned}$$

To calculate the average downloading time for a peer in steady state, we use Little's law [4] as follows:

$$\bar{x} = (\lambda - \theta\bar{x})T,$$

where T is the average downloading time and $\lambda - \theta\bar{x}$ is the average rate at which downloads are completed. Using (5), it is now easy to see that

$$T = \frac{1}{\beta} = \max\{\frac{1}{c}, \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})\}. \qquad (6)$$

Equation (6) provides several insights into the behavior of BitTorrent:

- The average downloading time T is not related to λ, the request arrival rate. Hence, the BitTorrent P2P system scales very well.

- When η increases, T decreases. This is because the peers share the file more efficiently.

- When γ increases, T increases because a larger γ means that there are fewer seeds in the system.

- Initially, when c increases, T decreases. However, once c is large enough ($\frac{1}{c} \leq \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$), increasing c further will not decrease T, because the downloading bandwidth is no longer the bottleneck. A similar observation can be made regarding the uploading bandwidth μ.

- While the average numbers of downloaders and seeds ($\bar{x}$ and $\bar{y}$) are functions of θ, the downloader leaving rate, the average downloading time T is independent of θ. The intuition behind this is as follows: when a downloader aborts its file transfer, it reduces the number of downloaders competing for resources in the system. On the other hand, it also reduces the number of peers that are uploading the file in the system. These two effects precisely cancel each other out in the fluid model leaving the average download time independent of θ.

- It is often true that the downloading bandwidth c of a peer would be much higher than its uploading bandwidth μ. Common examples of such an asymmetry are DSL and cable modem connections. For performance analysis purposes, it may be tempting to set $c = \infty$ as in [11, 24]. However, the expression for T in (6) shows that the average download time is not always constrained by the uploading bandwidth of the peers. In fact, if the seed leaving rate γ is smaller than μ, then the downloading bandwidth c determines the network performance even though c may be much larger than μ.

We briefly comment on the case $\eta = 0$, which means that downloaders do not upload data to each other and only download from seeds. If $\gamma < \mu$, the previous analysis as in the case of $\eta > 0$ still holds and $T = 1/c$. On the other hand, if $\gamma > \mu$, from (1), we can see

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} \leq (\mu - \gamma)y(t).$$

This tells us that $y(t)$ decreases at least exponentially. So if $\gamma > \mu$, the number of seeds will exponentially decrease to zero and the system dies. Recall that when $\eta > 0$, the system reaches a steady state no matter what γ is. So, it is very important for the downloaders to upload data to each other. Even if the file sharing is not very efficient (a small η), it can play an important role in keeping the system alive. From (6), we also see that η is important to the network performance. In the next subsection, we will derive an expression for η and argue that η is very close to 1 in BitTorrent.

3.2 Effectiveness of File Sharing

In this section, we present a simple model to calculate the value of η, which indicates the effectiveness of the file sharing. For a given downloader i, we assume that it is connected to $k = \min\{x-1, K\}$ other downloaders, where x is the number of downloaders in the system and K is the maximum number of downloaders to which a peer can connect. We also assume that each downloader has the information about which pieces the connected peers have. Hence if peer i has pieces that are of interest to at least one peer that is connected to it, then peer i will upload data. We then have

$$\eta = 1 - \mathbb{P}\left\{ \begin{array}{c} \text{downloader } i \text{ has no piece that} \\ \text{the connected peers need} \end{array} \right\}.$$

We assume that the piece distributions between different peers are independent and identical. Then

$$\eta = 1 - \mathbb{P}\left\{ \begin{array}{c} \text{downloader } j \text{ needs no} \\ \text{piece from downloader } i \end{array} \right\}^k,$$

where j is a downloader connected to i.

For each downloader, we assume that the number of pieces it has is uniformly distributed in $\{0, \cdots, N-1\}$, where N is the number of pieces of the served file. Let n_i denote the random variable describing the number of pieces at downloader i. We assume that given n_i, these pieces are chosen randomly from the set of all pieces of the file. This is a reasonable assumption because BitTorrent takes a rarest first piece selection policy when downloading. Under these assumptions, we have

$$\begin{aligned} & \mathbb{P}\left\{ \begin{array}{c} \text{downloader } j \text{ needs no} \\ \text{piece from downloader } i \end{array} \right\} \\ = \; & \mathbb{P}\{j \text{ has all pieces of downloader } i\} \\ = \; & \sum_{n_j=1}^{N-1} \sum_{n_i=0}^{n_j} \frac{1}{N^2} \mathbb{P}\{j \text{ has all pieces of } i | n_i, n_j\} \\ = \; & \sum_{n_j=1}^{N-1} \sum_{n_i=0}^{n_j} \frac{1}{N^2} \frac{\binom{N-n_i}{n_j-n_i}}{\binom{N}{n_j}} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{n_j=1}^{N-1} \frac{1}{N^2} \frac{\binom{N+1}{n_j}}{\binom{N}{n_j}} \\
&= \sum_{n_j=1}^{N-1} \frac{N+1}{N^2(N+1-n_j)} \\
&= \frac{N+1}{N^2} \sum_{n_j=1}^{N-1} \frac{1}{N+1-n_j} \\
&= \frac{N+1}{N^2} \sum_{m=2}^{N} \frac{1}{m} \approx \frac{\log N}{N}
\end{aligned}
$$

and

$$\eta \approx 1 - \left(\frac{\log N}{N}\right)^k. \tag{7}$$

Now, we will interpret the expression for realistic file sizes. In BitTorrent, each piece is typically $256KB$. Thus, for a file that is a few hundreds of megabytes in size, N is of the order of several hundreds. Hence, even if $k = 1$, η is very close to one. For BitTorrent, k is actually larger, since the maximum number of connections K is typically 40. This tells us that BitTorrent is very efficient in sharing files. When k increases, η also increases but very slowly and the network performance increases slowly. Note that, since k depends on the number of other peers in the system, it may be related to the arrival rate λ. Hence, when λ increases, the network performance increases but very slowly. Thus, our observation in the previous subsection that the network performance is essentially independent of λ still holds. This also matches the observations of real BitTorrent networks presented in [11, 24]. Note that when $k = 0$, the downloader is not connected to any other downloaders and hence $\eta = 0$.

3.3 Local Stability

When deriving the steady-state quantities $\bar{x}$, $\bar{y}$ and T, we implicitly assumed that the system is stable and will reach its equilibrium. In this section, we study the stability of the fluid model (1) around the equilibrium $\{\bar{x}, \bar{y}\}$.

When $\frac{1}{c} < \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$, the uploading bandwidth is the constraint and around a small neighborhood of $\{\bar{x}, \bar{y}\}$, we have

$$
\begin{aligned}
\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \lambda - \theta x(t) - \mu(\eta x(t) + y(t)) \\
\frac{\mathrm{d}y(t)}{\mathrm{d}t} &= \mu(\eta x(t) + y(t)) - \gamma y(t).
\end{aligned}
$$

Let

$$\mathbf{A}_1 = \begin{bmatrix} -(\mu\eta + \theta) & -\mu \\ \mu\eta & -(\gamma - \mu) \end{bmatrix}. \tag{8}$$

Then the eigenvalues of $\mathbf{A}_1$ determine the stability of the equilibrium $\{\bar{x}, \bar{y}\}$. Let ψ be an eigenvalue of $\mathbf{A}_1$. The eigenvalues of $\mathbf{A}$ are the solutions of

$$\psi^2 + (\mu\eta + \theta + \gamma - \mu)\psi + \mu\eta\gamma + \theta(\gamma - \mu) = 0. \tag{9}$$

Since $\frac{1}{c} < \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$, we have $\gamma > \mu$. When $\eta > 0$, both $\mu\eta + \theta + \gamma - \mu$ and $\mu\eta\gamma + \theta(\gamma - \mu)$ are greater than zero. So the eigenvalues have strictly negative real parts and the system is stable.

Similarly, when $\frac{1}{c} > \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$, the downloading bandwidth is the constraint and around a small neighborhood of $\{\bar{x}, \bar{y}\}$, we have

$$
\begin{aligned}
\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \lambda - \theta x(t) - cx(t) \\
\frac{\mathrm{d}y(t)}{\mathrm{d}t} &= cx(t) - \gamma y(t).
\end{aligned}
$$

Let

$$\mathbf{A}_2 = \begin{bmatrix} -(\theta + c) & 0 \\ c & -\gamma \end{bmatrix}. \tag{10}$$

Then the eigenvalues of $\mathbf{A}_2$ satisfy

$$\psi^2 + (\theta + \gamma + c)\psi + (\theta + c)\gamma = 0. \tag{11}$$

Again, since both $\theta + \gamma + c$ and $(\theta + c)\gamma$ are greater than zero, we see that the eigenvalues have strictly negative real parts and the system is stable.

The case where $\frac{1}{c} = \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$ is a little more tricky since the dynamics are determined by the matrix $\mathbf{A}_1$ or the matrix $\mathbf{A}_2$, depending upon the direction in which the system is perturbed. Thus, a linear analysis will not suffice to even determine local stability. To avoid lengthy arguments, we do not consider this special case here.

Even in the cases where $\frac{1}{c} \neq \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$, the global stability of the fluid model (1) may be hard to analyze because of the fact that the dynamics of the system changes depending upon whether $cx > \mu(\eta x + y)$ or not. Such systems are called switched linear systems; we refer the reader to the survey in [16] for the stability issues associated with such models.

3.4 Characterizing Variability

When the request arrival rate is large (which also means a large number of downloaders and seeds), the fluid model is a good approximation of the real system. However, it is important to understand how the number of seeds and downloaders vary around the numbers predicted by the deterministic model. In this subsection, we present a simple characterization of the variance of x and y around $\bar{x}$ and $\bar{y}$ using a Gaussian approximation.

To calculate the variance around the fluid limit, we make a few assumptions regarding the peer arrival process, the downloader and seed departure processes. We assume that peers arrive according to a Poisson process, and we assume that each downloader independently aborts its download after a certain amount of time which is exponentially distributed with mean $1/\theta$. Each seed stays in the system for a random time which is exponentially distributed with mean $1/\gamma$. Finally, to obtain a Markovian description of the system, we assume that the probability that some downloader becomes a seed in a small interval δ is given by $\min(cx, \mu(\eta x + y))\delta$. These assumptions can be easily relaxed to allow more general distributions for all the random variables involved by using phase-type distributions as in [14, 22, 10]. Under these assumptions, the number of downloaders and seeds at any time t can be described by

$$x(t) + \sqrt{\lambda}\hat{x}(t), \quad y(t) + \sqrt{\lambda}\hat{y}(t),$$

respectively, where $\hat{\mathbf{X}} = (\hat{x}, \hat{y})^T$ is the solution to the following stochastic fluid differential equation whose solution is known as the Ornstein-Uhlenbeck process:

$$\mathrm{d}\hat{\mathbf{X}}(t) = \mathbf{A}\hat{\mathbf{X}}(t)\mathrm{d}t + \mathbf{B}\mathrm{d}\mathbf{W}(t). \tag{12}$$

In (12), the components of $\mathbf{W}$ are independent standard Wiener processes (Brownian motions), with the entries of $\mathbf{A}$ and $\mathbf{B}$ being determined by whether the downloading or the uploading bandwidth is the bottleneck. Specifically, $\mathbf{A} = \mathbf{A}_1$ given by (8) if $\frac{1}{c} < \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$ and $\mathbf{A} = \mathbf{A}_2$ given by (10) if $\frac{1}{c} > \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$. In both cases, we have

$$\mathbf{B} = \begin{bmatrix} 1 & -\sqrt{\rho} & -\sqrt{(1-\rho)} & 0 \\ 0 & 0 & \sqrt{(1-\rho)} & -\sqrt{(1-\rho)} \end{bmatrix} \tag{13}$$

where $\rho := \frac{\theta}{\theta+\beta}$. We do not consider the more complicated case $\frac{1}{c} = \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})$ which is unlikely to occur in practice.

From (12), it is easy to compute the steady-state covariance of $\hat{\mathbf{X}}$, i.e, $\mathbf{\Sigma} = \lim_{t\to\infty} E(\hat{\mathbf{X}}(t)\hat{\mathbf{X}}^T(t))$. This is given by the so-called Lyapunov equation [3]

$$\mathbf{A\Sigma} + \mathbf{\Sigma A}^T + \mathbf{BB}^T = 0. \tag{14}$$

The steady-state variance of $\hat{x}$ is then given by $(1,1)$ element of $\mathbf{\Sigma}$ and the steady-state variance of $\hat{y}$ is given by the $(2,2)$ element of $\mathbf{\Sigma}$. The above result essentially states that, in steady-state, the number of seeds and downloaders is distributed as Gaussian random variables whose variances are determined by $\mathbf{\Sigma}$.

The formal proof required to establish (12) is beyond the scope of this paper. We will simply state here that it involves showing that the original stochastic process converges to the deterministic and stochastic differential equation limits when the arrival rate goes to ∞. This can be established using weak-convergence theorems such as the ones in [5, 12, 23].

4 Incentive Mechanism

In this section, we discuss the algorithm in BitTorrent which is intended to discourage free-riding. We first describe the algorithm and then study the optimal selfish behavior of the users under this algorithm.

4.1 Peer Selection Algorithm

There is a built-in incentive mechanism in BitTorrent to encourage users to upload. The basic idea is that each peer uploads to n_u peers from which it has the highest downloading rates (the default value of n_u is 4). But since a peer only has partial information of the whole network (i.e., it doesn't have the upload rate information of all peers), optimistic unchoking [8] is used to explore the network. In this section, our objective is to understand how the built-in incentive mechanism affects the network performance. Hence, we ignore the details of optimistic unchoking and assume that each peer has the global information of uploading rates. We also assume that there are no downloading bandwidth constraints, all peers are fully connected and have demands from each other.

Under the above assumptions, we can simplify the peer selection algorithm of BitTorrent as follows. We first sort the peers according to their uploading bandwidth (it could be the physical uploading bandwidth or the uploading bandwidth that has been set manually by the user) such that the first peer has the highest uploading bandwidth. If two or more peers have the same uploading bandwidth, they are randomly ordered. The peer selection process proceeds in steps with peer i choosing peers to upload at step i. In the real BitTorrent, the peer selection does not proceed in steps like this. However, after we describe the selection algorithm, it would be clear that the step-by-step selection process does not change the selection of the peers significantly. Let N be the total number of peers and let μ_i be the uploading bandwidth of peer i. Then at step i, peer i selects peers to upload according to the following rules.

1. If peer i is selected by peer j ($j < i$), then i selects j. For any peer k ($k \geq i$), let n_k^i be the number of peers that have selected peer k prior to step i.

2. If $n_i^i < n_u$ and $n_u - n_i^i \leq N - i$, peer i selects $n_u - n_i^i$ peers from the set $\{k | k > i\}$ using the following set of rules to prioritize a peer, say $k1$, over another peer $k2$:

   (a) If $\mu_{k1} > \mu_{k2}$, select $k1$.

   (b) If $\mu_{k1} = \mu_{k2}$ and $n_{k1}^i < n_{k2}^i$, select $k1$.

   (c) If $\mu_{k1} = \mu_{k2}$, $n_{k1}^i = n_{k2}^i$, and $k1 < k2$, select $k1$.

3. If $n_i^i < n_u$ and $n_u - n_i^i > N - i$, peer i selects all peers in $\{k | k > i\}$ and also randomly selects $(n_u - n_i^i) - (N - i)$ peers from the peers that i has not selected yet.

These rules are easy to understand. Rule 1 states that if the downloading rate from peer j to peer i is greater than or equal to the uploading rate of i, peer i should upload to peer j to try to keep the downloading rate high. We will show in Lemma 1 that $n_i^i \leq n_u$. So rule 1 will not violate the requirement that the number of uploads cannot exceed n_u. Rule $2(a)$ simply gives priority to peers with higher uploading rates. Rule $2(b)$ tries to treat peers with the same uploading rate as fairly as possible and rule $2(c)$ is simply a tie-break rule. Rule 3 takes care of the last several peers and makes sure that all peers have n_u uploads. The following lemma is a simple property of the peer selection algorithm.

Lemma 1 *With the peer selection algorithm, when peer i selects uploading peers, we have $n_i^i \leq n_u$ and for any $k2 > k1 \geq i$, $n_{k2}^i \leq n_{k1}^i \leq n_u$.*

Proof: First, when $i = 1$, $n_i^i = 0 \leq n_u$ and $n_{k2}^i = n_{k1}^i = 0 \leq n_u$, the lemma is true. Now, we assume that the lemma is true for peer i and prove that it is also true for peer $i + 1$ and hence by induction, it is true for all i.

If the lemma is true for i, we will have $n_{i+1}^i \leq n_i^i \leq n_u$ and $n_{k2}^i \leq n_{k1}^i \leq n_u$ for any $k2 > k1 \geq i + 1$. Now, if $n_i^i = n_u$, then peer i already has n_u uploads and it will not select any peer from $\{k | k > i\}$. Hence, for any $k > i$, $n_k^i = n_k^{i+1}$ and the lemma is true for $i + 1$. If $n_i^i < n_u$, then $n_{i+1}^i < n_u$. So, no matter whether peer i selects peer $i + 1$ or not, we always have $n_{i+1}^{i+1} \leq n_u$. To show the second part of the lemma, if $n_{k1}^i > n_{k2}^i$, after peer i makes the selection, we always have $n_{k1}^{i+1} \geq n_{k2}^{i+1}$. If $n_{k1}^i = n_{k2}^i$, according to rule 2, we also have $n_{k1}^{i+1} \geq n_{k2}^{i+1}$. Hence the lemma is true for $i + 1$. ■

Now let D_i be the set of peers that select peer i. We exclude peers that randomly select i by using rule 3 here for two reasons. First, each peer i has about equal chance to be selected and hence on average, the effect of the random selection can be equivalently seen as each peer getting a constant download rate d_r. Secondly, if the number of peers is large, d_r will be very small and can be ignored. The aggregate downloading rate of peer i then is

$$d_i = \frac{1}{n_u} \sum_{k \in D_i} \mu_k.$$

Note that if two peers have the same uploading bandwidth, they may get different downloading rates. Generally, if $\mu_i = \mu_{i+1} = \cdots = \mu_j$ are peers with the same uploading bandwidth, we will have $d_i \geq d_{i+1} \geq \cdots \geq d_j$. So, for a given peer i, the downloading rate not only depends on the uploading bandwidth μ_i, but also depends on how the peer is ordered with regards to other peers with the same uploading bandwidth. To eliminate the ambiguity, when there are two or more peers with the same uploading bandwidth μ, we define the downloading rate of these peers to be

$$d(\mu) = \frac{1}{j - i + 1} \sum_{k=i}^{j} d_k, \tag{15}$$

where i (resp. j) is the first (resp. last) peer with uploading bandwidth μ. Moreover, we have the following lemma when $n_u \geq 2$.

Lemma 2 *Suppose that peers $i, i+1, \cdots, j$ have the same uploading bandwidth μ, where i (resp. j) is the first (resp. last) peer with uploading bandwidth μ. If $j + i - 1 > n_u \geq 2$, then for any $k > j$, we have*

1. *$d_i \geq d_{i+1} \geq \cdots \geq d_j \geq d_k$,*

2. *$d_i > d_k$,*

3. *$d(\mu) > d_k$.*

Proof: First, from the peer selection rules, it is easy to see that for any two peers $k1 < k2$, $d_{k1} \geq d_{k2}$. So condition 1 is obviously true. Now, to prove condition 2, we only need to prove $d_i > d_{j+1}$. When peer i selects peers, if $n_i^i = n_u$ (i.e., peer i has already been selected by n_u peers that have uploading bandwidth greater than μ), then $d_i > \mu$. If $n_i^i < n_u$, peer i will select $n_u - n_i^i$ peers from $i+1, \cdots, j$. So, we always have $d_i \geq \mu$.

Now, when peer m ($i \leq m \leq j - n_u$) selects peers, if $n_{m+1}^m \geq 1$, obviously we will have $n_{m+1}^{m+1} \geq 1$. If $n_{m+1}^m = 0$, since peers m and $m+1$ have the same uploading bandwidth, from peer selection rule $2(b)$, we have $n_m^m \leq 1 < n_u$ and m will select peer $m+1$. Hence $n_{m+1}^{m+1} = 1$. In both case, we have $n_{m+1}^{m+1} \geq 1$.

When $m+1$ selects peers, since $n_{m+1}^{m+1} \geq 1$, $n_u - n_{m+1}^{m+1} \leq n_u - 1$. From $m+2$ to j, we have more than $n_u - 1$ peers with uploading bandwidth μ. So, $m+1$ will not select peer $j+1$ and peer $j+1$ can at most be selected by $n_u - 1$ peers with uploading bandwidth μ. So

$$d_{j+1} \leq \frac{1}{n_u}((n_u - 1)\mu + \mu_{j+2}) < \mu \leq d_i.$$

Since $d_{j+1} \geq d_k$ for any $k \geq j+1$, we have $d_i > d_k$.

From the condition 2 and the definition of $d(\mu)$ (15), it is easy to see that $d(\mu) > d_k$ and we are done. ■

Now, we have defined the peer selection rules. We will next study how these rules affect the peer's choice of μ_i, the uploading bandwidth.

4.2 Peer Strategy

The objective of the incentive mechanism is to encourage users to contribute. In BitTorrent, the uploading bandwidth can be chosen by each user up to a maximum of the physical uploading bandwidth. The purpose of the rest of this section is to study how the incentive mechanism will affect the peer strategy, i.e, how the users set their bandwidth. Let p_i be the physical uploading bandwidth of peer i and let $\{\mu_{-i}\}$ be the set of uploading bandwidth chosen by the peers except μ_i. Let $d_i(\mu_i, \mu_{-i})$ be the aggregate downloading rate of peer i when the uploading bandwidth of peer i is μ_i. When $\{\mu_{-i}\}$ is given, it is obvious that d_i is a non-decreasing function of μ_i. So when $\mu_i = p_i$, peer i gets the maximum downloading rate. But setting $\mu_i = p_i$ is not necessarily the best strategy for peer i. For each peer i, d_i is the gain and μ_i is the cost. A peer wants to maximize the gain, but at the same time, it also wants to minimize the cost. Here, we assume that maximizing the gain has priority over minimizing the cost. Intuitively, we may want peer i to choose μ_i such that

$$\mu_i = \min\{\tilde{\mu}_i | d_i(\tilde{\mu}_i, \mu_{-i}) = d_i(p_i, \mu_{-i})\}. \tag{16}$$

But unfortunately, the minimum of the set $\{\tilde{\mu}_i | d_i(\tilde{\mu}_i, \mu_{-i}) = d_i(p_i, \mu_{-i})\}$ may not exist (e.g., for the set $(4, 6]$). If we take this into account, the best strategy for peer i will be

$$\mu_i = \min\left\{\inf\{\tilde{\mu}_i | d_i(\tilde{\mu}_i, \mu_{-i}) = d_i(p_i, \mu_{-i})\} + \varepsilon, p_i\right\}, \tag{17}$$

where $\varepsilon > 0$ is a small number. The parameter ε can be interpreted as the small difference between two rates that a peer can differentiate. Note that even if the minimum of $\{\mu_i | d_i(\mu_i, \mu_{-i}) = d_i(p_i, \mu_{-i})\}$ exists, it is still better to add a small number ε. Because if the uploading bandwidth of two peers are very close, we may not be able to detect the difference between them. Hence, adding a small positive number can help differentiate peer i from other competing peers.

Given the peer selection algorithm (game rules), we can now study the system as a non-cooperative game. A Nash equilibrium for our problem is a set of uploading rates $\{\bar{\mu}_i\}$ such that

$$\bar{\mu}_i = \min\left\{\inf\{\tilde{\mu}_i | d_i(\tilde{\mu}_i, \bar{\mu}_{-i}) = d_i(p_i, \bar{\mu}_{-i})\} + \varepsilon, p_i\right\}.$$

Let's consider a small BitTorrent network with 6 peers. The number of uploads $n_u = 4$ for all peers. We will show that if the peers have different physical uploading bandwidth and the minimum uploading bandwidth $\min\{p_i\} > 2\varepsilon$, there is no Nash equilibrium point for the system. In this simple example, we can see that if the uploading bandwidth μ_i of peer i is less than those of all other peers, then peer i will get zero downloading rate because the other five peers will upload to each other and not to peer i. On the other hand, once μ_i is greater than the uploading bandwidth of at least one peer, peer i will get the same downloading rate even if $\mu_i < p_i$. So the strategy for peer i (17) in this example turns out to setting μ_i such that it is the fifth highest uploading bandwidth. Now, assume that there is a Nash equilibrium point $\{\bar{\mu}_i\}$ and we sort the peers by their uploading bandwidth such that $\bar{\mu}_1$ is the highest uploading bandwidth. Then we have $\bar{\mu}_5 > \bar{\mu}_6$. Otherwise, if they are equal, since the two peers have different physical uploading bandwidth, there is at least one peer with $\mu_i < p_i$ and this peer can increase its uploading bandwidth to increase its download rate. Now, if $\bar{\mu}_5 > \bar{\mu}_6$, we know that peer 6 gets a zero downloading rate. Since $\{\bar{\mu}_i\}$ is a Nash equilibrium, given $\{\bar{\mu}_{-6}\}$, the maximum downloading rate that peer 6 can get is also zero. Hence, from (17), we have $\bar{\mu}_6 = \varepsilon$. Now, if $\bar{\mu}_6 = \varepsilon$, from (17), we have $\bar{\mu}_5 = 2\varepsilon < \min\{p_i\}$. If $\bar{\mu}_5 < \min\{p_i\}$, peer 6 can increase its uploading bandwidth such that $\mu_6 > \bar{\mu}_5$, which contradicts the fact that $\bar{\mu}_5$ is the fifth highest uploading bandwidth. Hence, there is no Nash equilibrium point for the system. While there may be no Nash equilibrium point for a general network setting, when the network consists of groups of peers where members of each group have the same uploading and downloading bandwidths, there does exist a Nash equilibrium point as we will show in the next subsection.

4.3 Nash Equilibrium Point

We consider a network with a finite number of groups of peers. In group j, all peers have the same physical uploading bandwidth p_j. Note that this is in fact a good model for the current Internet users, who have only a finite number of network access methods (dial-up, dsl, cable modem, etc). Let g_j be the set of peers in group j and $||g_j||$ be the number of peers in group j. Without loss of generality, we also assume $p_1 > p_2 > \cdots$.

Proposition 1 *If $n_u \geq 2$ and the number of peers in a group $||g_j|| > n_u + 1$ for all groups, there exists a Nash equilibrium point for the system, in which $\bar{\mu}_i = p_j$ if peer $i \in g_j$. Moreover, with any initial setting of $\{\mu_i^0\}$, the system converges to the Nash equilibrium point $\{\bar{\mu}_i\}$.*

Proof: We first prove that $\{\bar{\mu}_i\}$ is a Nash equilibrium point. To prove this, we only need to prove that for any peer i, if $\mu_i < \bar{\mu}_i$, then $d_i(\mu_i, \bar{\mu}_{-i}) < d_i(\bar{\mu}_i, \bar{\mu}_{-i})$. Without loss of generality, we assume that $i \in g_j$. Since $||g_j|| > n_u + 1$, if we set $\mu_i < \bar{\mu}_i = p_j$, there will be still at least $n_u + 1$ peers with uploading bandwidth p_j. From Lemma 2, it is easy to see that $d_i(\mu_i, \bar{\mu}_{-i}) < d_i(\bar{\mu}_i, \bar{\mu}_{-i})$.

To prove convergence, we first consider the first group g_1. Let v^m be the $(n_u + 1)$th highest uploading bandwidth after m rounds of iterations. Then v^0 is the $(n_u + 1)$th highest uploading bandwidth of the initial set $\{\mu_i^0\}$. If after m rounds, $v^m + \varepsilon \leq p_1$, then in the $m + 1$ round, any peer $i \in g_1$ will increase its uploading bandwidth to $\mu_i \geq v^m + \varepsilon$ to maximize its downloading rate. Since $||g_1|| > n_u + 1$, after the $m + 1$ round, we will have $v^{m+1} \geq v^m + \varepsilon$. The increase in v^m will continue until $v^m = p_1$ and the peers cannot increase their uploading bandwidth anymore. In this case, any peer $i \in g_1$ will have the uploading bandwidth $\mu_i = p_1$. Once peers in the first group reach their maximum limit, they will not change their uploading bandwidth anymore. We can now use a similar argument to prove that peers in the second group will also reach the Nash equilibrium point. Continuing in a similar fashion, we can establish that the whole system converges to the Nash equilibrium point. ■

5 Optimistic Unchoking

In Section 4, we assume that each peer knows the uploading bandwidths of all other peers. In reality, each peer only has the rate information about peers from which it is downloading. Hence optimistic unchoking is used to explore the network and obtain information about other peers. In this section, we briefly study the effect of optimistic unchoking on free-riders. Specifically, while in Section 4.3, we showed that rational users would set their uploading rate to be equal to the maximum possible limit, here we will show that the maximum downloading rate that an irrational user who chooses to free-ride is limited to $\frac{1}{n_u+1}$ of the normal downloading rate that they can get if they behave rationally.

5.1 Free-Riding

Free-riding means that a peer does not contribute anything to the system, while it attempts to obtain service (or downloading) from other peers. If peers have global information, the free-riding problem can be solved by not uploading to peers with zero uploading bandwidth. In reality, peers use optimistic unchoking to explore the network and this gives an opportunity to free-ride. To illustrate it, let's consider a simple example.

We consider a network with a group of peers (g_1) that have the same uploading bandwidth μ. The number of peers in the group is N. We assume that each peer has n_u uploads and one optimistic unchoking upload. Now, a new peer j with zero uploading bandwidth joins the network. Each peer $i \in g_1$ will randomly choose a peer that it is not currently uploading to as the target of its optimistic unchoking. So, for peer i, on average, $\frac{1}{N-n_u}$ of the time, it will optimistically upload to peer j. Since there is a total of N peers in g_1, the total average downloading rate of peer j will be

$$N\frac{1}{N-n_u}\frac{\mu}{n_u+1} \approx \frac{\mu}{n_u+1},$$

when N is large.

In this example, we see that because of optimistic unchoking, peer j contribute nothing to the system, but it still get an average downloading rate of $\frac{\mu}{n_u+1}$. In current BitTorrent, $n_u = 4$ and thus a free-rider gets 20% of the possible maximum downloading rate. It would seem that n_u can be increased to reduce the amount that a free-rider can get. However, choosing n_u to be large means that multiple TCP connections have to share the same bandwidth and thus may lead to more time-outs and result in poor performance. The choice of an optimal n_u or other methods to alleviate the free-riding problem is a subject for further study.

6 Experimental Results

We performed a series of experiments to validate the fluid model described in Section 3. In the first two experiments, we compare a simulated BitTorrent-like network and the fluid model. In the last experiment, we actually introduced a seed into the BitTorrent network, studied the evolution of the seeds/downloaders, and compared it to our fluid model results. Due to copyright reasons, we obviously could not introduce a very popular file into the network. However, as we will show in our experimental results, even for a file which had a total of less than 100 completed downloads, the match between the fluid model and the observed data is quite close.

6.1 Experiment 1

In Figs 1 and 2, we compare the simple deterministic fluid model that we derived with the results from a discrete-event simulation of a BitTorrent-like network. In the discrete-event simulation, we use the Markov model described in Section 3.4. We chose the following parameters for this simulation: $\mu = 0.00125$, $c = 0.002$, $\theta = \gamma = 0.001$. When the number of downloaders is 1, we set $\eta = 0$, otherwise, we set $\eta = 1$. This is in keeping with our observation regarding the efficiency of the download as described in Section 3.2. Initially, there is one seed and no downloader. We also keep the number of seeds no less than one during the entire simulation. We change the arrival rate λ from 0.04 to 40 and plot number of seeds/downloaders normalized by the arrival rate, i.e., $\frac{y(t)}{\lambda}$ and $\frac{x(t)}{\lambda}$, from both simulations and the fluid model. In this experiment, since $\gamma < \mu$, we know that downloading bandwidth is the bottleneck. From the figures, we see that the simple fluid model is a good approximation of the system when λ is large, but the match is quite good even for small λ. The figures also indicate that the number of downloaders increases linearly with the arrival rate λ. By Little's law, this implies that the average download time is constant, independent of the peer arrival rate, which shows that the system scales very well. In other

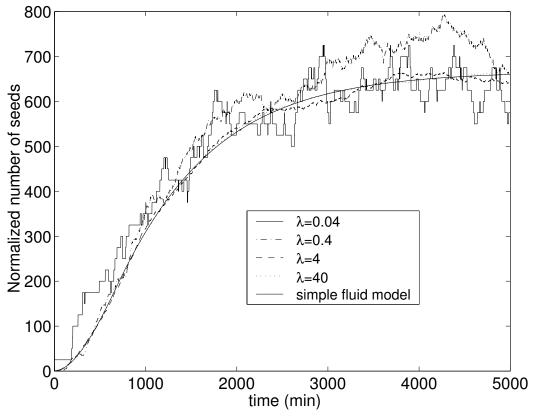

Figure 1: Experiment 1 : The evolution of the number of seeds as a function of time

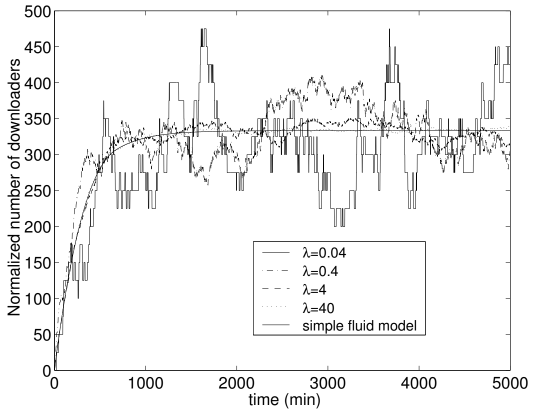

Figure 2: Experiment 1 : The evolution of the number of downloaders as a function of time

words, even very popular files can be downloaded at the same speed as less popular files.

6.2 Experiment 2

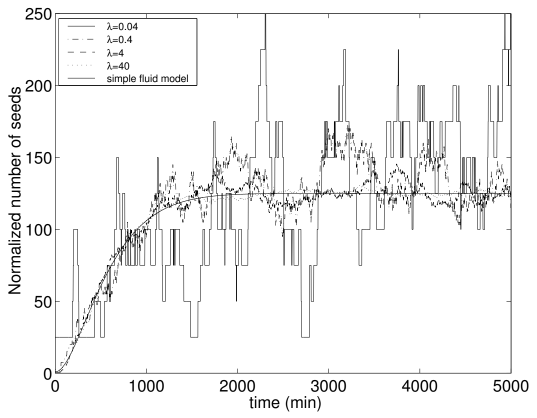

Figure 3: Experiment 2 : The evolution of the number of seeds

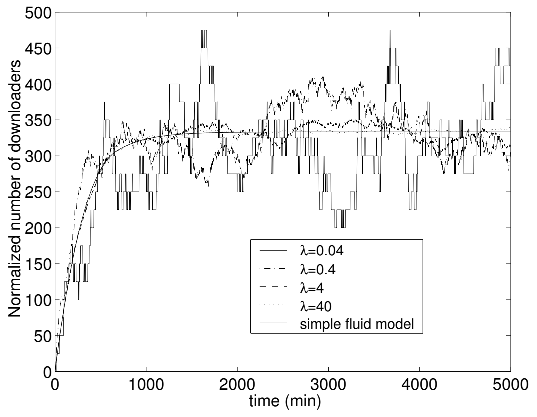

Figure 4: Experiment 2 : The evolution of the number of downloaders

In Figs. 3 and 4, we have the same setting as the first experiment, except that now we set $\gamma = 0.005$. With the change of γ, the uploading bandwidth now becomes the bottleneck. In this setting, we have the similar result as before. Again, we see that the simple fluid model is accurate when λ is large, but performs well even for smaller λ. We also plot the histogram of

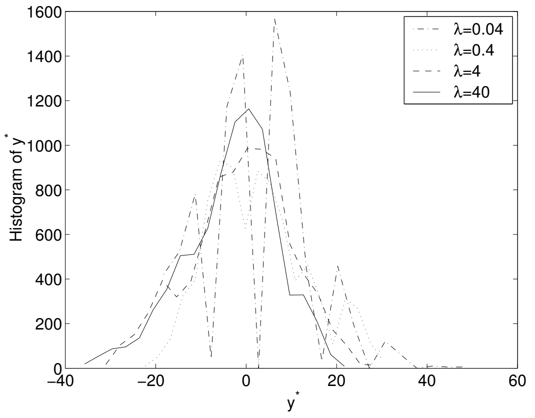

Figure 5: Experiment 2 : Histogram of the variation of the number of seeds around the fluid model

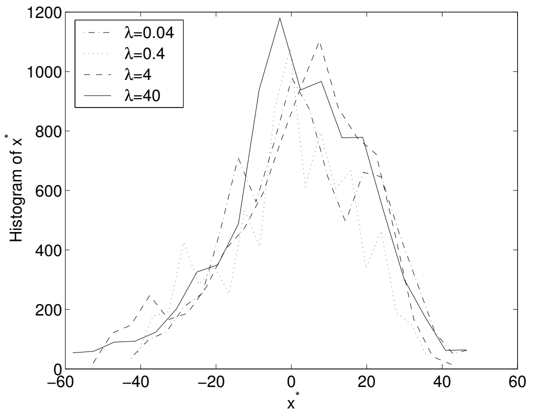

Figure 6: Experiment 2 : Histogram of the variation of the number of downloaders around the fluid model

$\hat{x}$ and $\hat{y}$ in Figs. 5 and 6,

$$\hat{x}(t) = \frac{x_{sim}(t) - x(t)}{\sqrt{\lambda}}$$

and

$$\hat{y}(t) = \frac{y_{sim}(t) - y(t)}{\sqrt{\lambda}},$$

where $x_{sim}(t)$ and $y_{sim}(t)$ are the number of downloaders and seeds respectively in the actual simulation and $x(t)$ and $y(t)$ are the number of downloaders and seeds in deterministic fluid model. From the theory presented in Section 3.4, we expect the histograms to look roughly Gaussian and this fact is borne out by the figures for sufficiently large λ. We can see that the variance of $\hat{x}$ and $\hat{y}$ do not change much when λ changes from 0.04 to 40.

6.3 Experiment 3

In this experiment, we introduced a file into the BitTorrent network and collected the log files of the BitTorrent tracker for a time period of around three days. When a peer joins/leaves the system or completes the download, it reports the event to the tracker. In addition, peers regularly report information such as the total amount of data uploaded/downloaded so far, the number of bytes that still need to be downloaded, etc. The tracker keeps all the information in the log files. Hence, we can analyze the tracker log files and retrieve useful information. The parameters λ, θ, and γ can be measured by counting the peer arrival, the downloader departure, and the seed departure respectively. However, from the tracker log files, we cannot determine whether the uploading bandwidth or the downloading bandwidth is the bottleneck. So we assume the uploading bandwidth is the bottleneck and estimate μ by dividing the measured total uploading rate by the number of peers (i.e., we assume that $\eta = 1$). The size of the file that was introduced was around $530MB$. The average uploading bandwidth was estimated to be $90kb/s$. We use 1 min as the time unit to calculate arrival rates, departure rates, etc. The normalized uploading bandwidth (normalized by the file size in bytes) was estimated $\mu = 0.0013$. The downloader leaving rate was estimated to be $\theta = 0.001$. An interesting feature that we observed in the real BitTorrent is that λ and γ are in fact time-varying. We attribute this to the fact that when a new file is introduced into the system, the first few seeds stay in the system long enough to ensure that there is a sufficient population of peers to sustain the system. If the initial seeds depart too quickly, the system will simply die, i.e., there will be no one to download from.

From the tracker logs, we estimate that, for $t \leq 800min$, $\lambda = 0.06$ and $\gamma = 0.001$. When $t \geq 1300min$, $\lambda = 0.03$ and $\gamma = 0.0044$. In between, the arrival rate increases roughly linearly. In our fluid model simulation, for time between $800min$ and $1300min$, we let λ and γ change linearly. We also set the downloading bandwidth $c = 1$ for the fluid model simulation (note that the actual value of c will not affect the fluid model results if it is above a certain threshold).

The simulation results are shown in Figs 7 and 8.

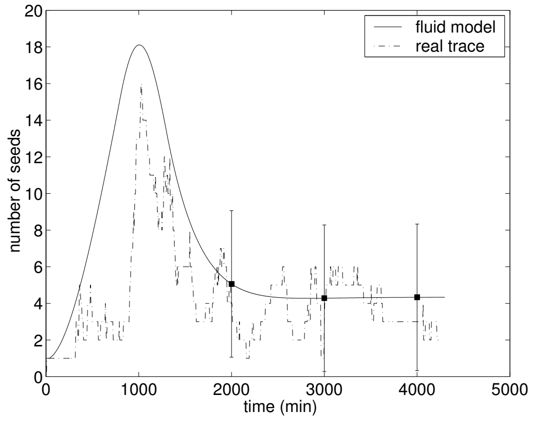

Figure 7: Experiment 3 : Evolution of the number of seeds

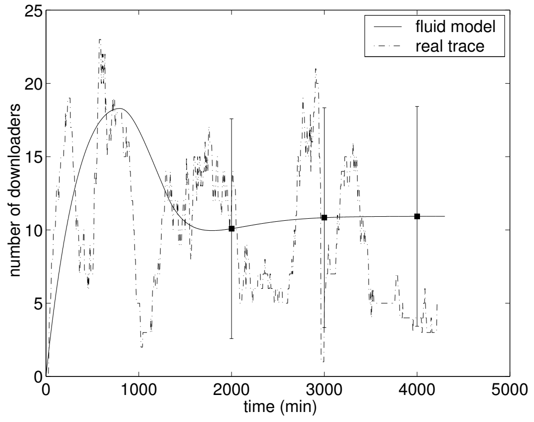

Figure 8: Experiment 3 : Evolution of the number of downloaders

The real trace is measured from the tracker log file and the fluid model is calculated by using the above measured parameters. For the fluid model, we also numerically calculate the standard deviation from the steady state network parameters by using (14) and plot the error bar for 95% confidence intervals. From Fig. 7, we see that the fluid model captures the evolution of the number of seeds well. In Fig. 8, the oscillation of the number of downloaders is more significant. This is because that the file is not very popular and the arrival rate λ is small. Hence, our model is only an approximation of the real network. But despite this, we can see that the oscillation is within the level suggested by the 95% confidence interval.

7 Conclusions

In this paper, we first presented a simple fluid model for BitTorrent-like networks and studied the steady-state network performance. Specifically, we obtained expressions for the average number of seeds, the average number of downloaders, and the average downloading time as functions of the peer arrival rate, downloader leaving rate, seed leaving rate, uploading bandwidth, etc, which explicitly give us insight on how the network performance is affected by different parameters. We also characterized the variability of the system by applying limit theorems to the stochastic model when the arrival rate is large. We then abstracted the built-in incentive mechanism of BitTorrent and studied its effect on network performance. Under certain conditions, we proved that a Nash equilibrium exists, under which each peer chooses its physical uploading bandwidth to be equal to the actual uploading bandwidth. We also briefly discussed the effect of optimistic unchoking on free-riding. Our experimental results show that the simple fluid model can capture the behavior of the system even when the arrival rate is small.

References

[1] Entropia. http://www.entropia.com.

[2] Top applications (bytes) for subinterface: SD-NAP traffic, 2002. www.caida.org/analysis/workload/byapplication/sdnap.

[3] L. Arnold. *Stochastic Differential Equations: Theory and Applications.* John Wiley, New York, NY, 1974.

[4] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[5] P. Billingsley. *Convergence of Probability Measures.* Wiley, 1968.

[6] F. Clevenot and P. Nain. A Simple Fluid Model for the Analysis of SQUIRREL. Technical report, INRIA RR-4911, 2003.

[7] F. Clevenot, P. Nain, and K. Ross. Stochastic Fluid Models for Cache Clusters. Technical report, INRIA RR-4815, 2003.

[8] B. Cohen. Incentives build robustness in bittorrent, May 2003. http://bitconjurer.org/BitTorrent /bittorrentecon.pdf.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[10] A. Das and R. Srikant. Diffusion approximations for a single node accessed by congestion-controlled sources. *IEEE Transactions on Automatic Control*, 45(10):1783–1799, October 1998.

[11] G. de Veciana and X. Yang. Fairness, incentives and performance in peer-to-peer networks. In *the Forty-first Annual Allerton Conference on Communication, Control and Computing, Monticello, IL*, Oct. 2003.

[12] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and Convergence.* Wiley, 1994.

[13] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *Proc. of Infocom*, 2003.

[14] P. W. Glynn. On the Markov property of the $GI/G/\infty$ Gaussian limit. *Advances in Applied Probability*, 14:191–194, 1982.

[15] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC '02)*, Monterey, California, 2002.

[16] D. Liberzon and A. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, pages 59–70, 1999.

[17] T. S. Eugene Ng, Y.-H. Chu, S. G. Rao, K. Sripanidkulchai, and Hui Zhang. Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-To-Peer Systems. In *INFOCOM*, 2003.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[19] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network, 2001.

[20] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[21] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishman. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[22] W. Whitt. On the heavy-traffic limit theorems for $GI/G/\infty$ queues. *Advances in Applied Probability*, 14:171–190, 1982.

[23] W. Whitt. *Stochastic Process Limits.* Springer, 2002.

[24] X. Yang and G. de Veciana. Service Capacity of Peer to Peer Networks. In *INFOCOM*, 2004.