Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 29 (2006) 3946–3956
computer communications
www.elsevier.com/locate/comcom

Analyzing the performance and fairness of BitTorrent-like networks using a general fluid model

Yao Yue [a,*], Chuang Lin [a], Zhangxi Tan [b]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[b] Department of Electronic Engineering and Computer Science, University of California, Berkeley, USA

Received 28 March 2006; received in revised form 9 June 2006; accepted 15 June 2006
Available online 18 July 2006

Abstract

In this paper, a general fluid model is developed to study the performance and fairness of BitTorrent-like networks. The fluid model incorporates two important features previously isolated from system performance models, user settings with multiple groups and inter-group data exchange induced by the choking algorithm. Our numerical results point out some key parameters of performance, such as the staying time of seeders. Generally, selfish behavior does not receive equal performance degradation, and in some scenarios users have strong incentives of free-riding. We also find content delivery can be greatly deterred when malicious free-riders are overwhelming.

Keywords: BitTorrent; Fluid model; Performance; Fairness

1. Introduction

Peer-to-peer (P2P) applications have become a major source of Internet traffic and are still on the rise [1]. As a typical P2P software and protocol, BitTorrent [2] has achieved a remarkable success since its release in 2003. It now takes over half of the P2P traffic in some areas [3] and offers an appealing target of P2P research. A BitTorrent session is the procedure of distributing a file to interested clients, called *peers*. Regarding BitTorrent sessions, four issues have been widely discussed on performance and fairness. The first is scalability, namely, how BitTorrent handles varied population. The second is the maintenance of a session. Without sufficient content providers around, a BitTorrent session quickly ceases. The third issue is the efficiency, i.e., how fast peers can finish downloading and how well their bandwidths are utilized. The fourth issue concerning BitTorrent is how well it deters selfish behavior, known as free-riding in most P2P literature. Due to the cooperative nature of BitTorrent-like networks, performance and fairness are interwoven and thus should not be treated separately.

Previous works on BitTorrent have taken either an experimental or a theoretical approach. Izal et al. [4] collected realistic data from the tracker's perspective.[1] Arnaud Legout and Urvoy-Keller [5] analyzed a peer's log files after participating in 12 BitTorrent sessions. Bharambe et al. [6] simulated and altered some major mechanisms in BitTorrent. On the other hand, in [7] a branching process model was used to estimate the service capacity of BitTorrent-like networks after startup. Qiu and Srikant [8] introduced a simple fluid model of identical users, and obtained some analytical solutions as well as the equilibrium strategy for peers. However, performance models in [7,8] failed to capture the influence of multiple groups and optimistic unchoking.

In this paper, we combine a multi-group setting and the choking algorithm in a general fluid model to analyze the

* Corresponding author. Tel.: +86 10 62772487; fax: +86 10 62771138.
E-mail address: yueyao@csnet1.cs.tsinghua.edu.cn (Y. Yue).

[1] Trackers help newcomers to find other peers, and aggregate peer statistics.

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.06.015

performance and fairness of BitTorrent. Multiple groups are necessary because an assumption of identical peers does not reflect realistic peer composition. The multi-group setting further enables and necessitates the study of *optimistic unchoking*, a mechanism via which different groups unavoidably exchange data. After incorporating both features, numerical solutions can be generated to reveal parameters essential to system performance and fairness. Our results show that group composition and the number of seeders are very important to overall performance. In particular, normal peers may get harmed if the session is dominated by malicious free-riders. Calculation also proves that peers with a smaller uploading capacity do not suffer an equally prolonged downloading time. For certain scenarios, the low-bandwidth peers will have strong incentive of free-riding even if they are performance-sensitive.

The rest of the paper is organized as follows. Section 2 provides a very brief description of the BitTorrent protocol. Section 3 generates a general fluid model through three steps. First multiple groups are introduced in a fluid model without data exchange between groups, followed by a discussion of its steady state. Second, assuming optimistic unchoking is running, how fast downloading from other leechers can converge to a steady rate and its value are computed for each group. Third, the steady state of the fluid model is revised to consider the influence of optimistic unchoking. Section 4 presents numerical results of the general model and carries some discussion about the key elements toward better efficiency and fairness. Section 5 concludes the paper.

2. Background

In BitTorrent, peers aiming at the same file form a large pool called a *swarm*, made up of both *leechers* (downloading peers) and *seeders* (uploading peers). A peer typically keeps TCP connections with at most 40–100 peers, which become its *neighbors*. To better cope with the TCP protocol, the number of leechers a peer actually uploads to at one time is only a few. The local peer holds back uploading to most of its neighbors by marking them as *choked*, otherwise they are *unchoked*. Neighbors in want of data available at the local peer are *interested*, others are *not interested*. The leechers receiving data, i.e., interested and unchoked leechers, are called *downloaders* by the local peer.

The *choking algorithm* locally run by a leecher decides the choking state of remote peers, which could be further divided into *regular unchoking* and optimistic unchoking. Regular unchoking periodically selects and unchokes neighbors who are offering the best uploading rates. On the other hand, as potentially better uploading rates could be provided by other neighbors, the local leecher tries them out using optimistic unchoking. In practice, the local peer transfers data to some randomly chosen neighbor for 30 s, if this rate is sufficiently high, the regular unchoking at the remote peer should start rewarding before the 30-s period runs out. Different from leechers, seeders require no reciprocal uploading and thus merely consider leechers' downloading capacity. The latest choking algorithm takes a step further on fairness and distributes a seeder's uploading chances more or less randomly among connections.

To get a complete file, peers join the corresponding session after downloading a small *.torrent* file containing meta information of the target and where peer lists can be requested. The actual file distributed via BitTorrent is split into much smaller, verifiable parts called *pieces* and even smaller request units called *blocks*. A leecher follows the *rarest first* rule, so that a piece with the least occurrence among neighboring peers is requested first. When a peer asks for a certain piece, it sends requests for all blocks within that piece before any other piece, known as the *strict priority* policy. Therefore, new leechers can start exchanging data with other peers almost immediately after they join.

3. A general fluid model

This section develops a general fluid model to facilitate studying important system indices like swarm population and average downloading time. Here, we first introduce multiple leecher groups into the model without inter-group data exchange. Then, after discussing the impact of optimistic unchoking in detail, the fluid model is revised to take that into account.

3.1. Introducing multiple groups

To capture the disparity inside a swarm, the uni-group fluid model in [8] is extended by introducing a number of groups. In the general model, peers belonging to the same group are characterized by the same set of parameters, probably varying among groups. Previous work has presented expressions of the leecher/seeder population and the average downloading time under the assumption of identical peers [7,8]. Nevertheless, in a real BitTorrent session seldom do all peers possess the same physical bandwidths. Moreover, peers can effortlessly limit the uploading/downloading bandwidth in many BitTorrent clients. This could be appealing to some DSL users, for their uploading bandwidths are much more constrained than downloading bandwidths.

In our general model, it is presumed that only finite number of groups exist. This presumption is rationalized by the finite access methods of today's Internet users. The file size of each session is defined as 1, and the bandwidths are normalized accordingly. Since our model focuses on the overall performance of a downloading session, the temporary leaving of re-joining of peers is not explicitly modeled. For a swarm of n peer groups, variables and parameters describing the model at time t are listed as following:

λ_i	Arriving rate of G_i leechers. Peers arrive according to the Poisson process
θ_i	The rate a G_i leecher aborts the session
γ	The rate a seeder leaves the session. Since leaving is manually done by users, it is assumed all groups have the same value
μ_i	Uploading bandwidth limit of a G_i leecher
c_i	Downloading bandwidth limit of a G_i leecher, $c_i \geqslant \mu_i$
$x_i(t)$	Number of leechers that belong to group G_i at time t
$y(t)$	Number of seeders at time t
$\mu(t)$	Average uploading bandwidth of seeders
η	File sharing efficiency, defined as the probability a leecher has at least one piece requested by its neighbors. Qiu and Srikant [8] argue η is very close to 1 in BitTorrent

The variables $x_i(t)$, $y(t)$, and $\mu(t)$ are decided by the parameters λ_i, θ_i, γ, μ_i, c_i, and η. At time t, the size of leecher group i changes by the margin between the arriving and departing rates. The arriving rate of group i is set to a system parameter λ_i, the abandoning rate of leechers is $\theta_i x_i$, the rate group i peers become seeders is $\min\{\mu_i\eta x_i(t) + \mu\rho_i(t)y(t),\ c_i x_i(t)\}$. The size and composition of the seeder group evolve similarly. $\mu(t)$ changes according to the composition of seeders. Hence, we obtain a set of differential equations:

$$\begin{aligned}
\frac{\mathrm{d}x_i}{\mathrm{d}t} &= \lambda_i - \theta_i x_i(t) - \min\{\mu_i\eta x_i(t) + \mu\rho_i(t)y(t), c_i x_i(t)\},\\
\frac{\mathrm{d}y}{\mathrm{d}t} &= \sum_{i=1}^{n} \min\{\mu_i\eta x_i(t) + \mu\rho_i(t)y(t), c_i x_i(t)\} - \gamma y(t),\\
\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} &= \sum_{i=1}^{n} \min\left\{\frac{\mu_i\eta x_i(t)}{y(t)} + \mu\rho_i(t), \frac{c_i x_i(t)}{y(t)}\right\}\mu_i - \gamma\mu(t).
\end{aligned} \tag{1}$$

The auxiliary variables in Eq. (1) are defined as: $x(t)$ is the total number of leechers at time $t, x(t) = \sum_{i=1}^{n}\mu_i x_i(t)$ and $\rho_i(t)$ is the proportional population of the ith group in all leechers, $\rho_i(t) = \frac{x_i(t)}{x(t)}$.

3.1.1. Steady state

After the startup period, a BitTorrent session enters a lasting steady state [4] whose performance and fairness concern the majority of users. Steady state is solved by letting $\frac{\mathrm{d}x_i(t)}{\mathrm{d}t} = \frac{\mathrm{d}y(t)}{\mathrm{d}t} = 0$ in Eq. (1). Variable values in this regime are represented as $\bar{x}_i$, $\bar{y}$, etc.

The following discussion will be based on a swarm of two groups, the simplest case without losing peer disparity and in which the fairness problem may occur. Similar analysis applies to swarms with more groups. To have a glimpse of the analytical solutions, we solve the most complex case where all leechers are limited by their uploading bandwidths. The assumption is equivalent to $\mu_i\eta\bar{x}_i + \bar{\mu}\bar{\rho}_i\bar{y} < c_i\bar{x}_i$ for $i = 1,2$. This case is important, because the lack of uploading bandwidth is common among peers and it points out when the protocol fails to make the best use of the network infrastructure. Thus, the steady-state equations can be simplified as:

$$\begin{aligned}
0 &= \lambda_1 - \theta_1\bar{\rho}\bar{x} - (\mu_1\eta\bar{\rho}_1\bar{x} + \bar{\mu}\bar{\rho}_1\bar{y}),\\
0 &= \lambda_2 - \theta_2\bar{\rho}_2\bar{x} - (\mu_2\eta\bar{\rho}_2\bar{x} + \bar{\mu}\bar{\rho}_2\bar{y}),\\
0 &= \mu_1\eta\bar{\rho}_1\bar{x} - \mu_2\eta\bar{\rho}_2\bar{x} + \bar{\mu}\bar{y} - \gamma\bar{y},\\
&\bar{\mu}[\mu_1\eta\bar{\rho}_1\bar{x} + \bar{\mu}_2\eta\bar{\rho}_2\bar{x} + \bar{\mu}\bar{y}]\\
&\quad = \mu_1(\mu_1\eta\bar{\rho}_1\bar{x} + \bar{\mu}\bar{\rho}_1\bar{y}) + (\mu_2\eta\bar{\rho}_2\bar{x} + \bar{\mu}\bar{\rho}_2\bar{y}).
\end{aligned} \tag{2}$$

As mentioned before, $\eta = 1$ can be treated as 1, then $\bar{\rho}_1$ $(0 < \bar{\rho}_1 < 1)$ turns out to be the root of this quadratic equation:

$$\begin{aligned}
0 = {} & \lambda_1(-\mu_2\theta_2 + \gamma\mu_2 + \gamma\theta_2) + [\lambda_1(2\mu_2\theta_2 - \gamma\mu_2 - \gamma\theta_2\\
& - \mu_1\mu_2 - \mu_1\theta_2 + \mu_1^2) + \lambda_2(\mu_2\theta_1 - \mu_1\mu_2 - \gamma\mu_1 - \mu_2^2\\
& - \gamma\theta_1)]z + [\lambda_1(\mu_1\theta_2 - \mu_2\theta_2 - \mu_1\mu_2 - \mu_1^2) + \lambda_2(\mu_1\theta_1\\
& - \mu_2^2 - \mu_2\theta_1 - \mu_1\mu_2)]z^2.
\end{aligned} \tag{3}$$

When $\bar{\rho}_1$ is known, $\bar{x}$, $\bar{y}$ can be expressed as follows, and other variable values can be calculated with ease.

$$\begin{aligned}
\bar{x} &= \frac{(\lambda_1 + \lambda_2)(\gamma - \bar{\mu})}{(\gamma - \bar{\mu})[\bar{\rho}_1\theta_1 + \bar{\rho}_2\theta_2] + \gamma[\bar{\rho}_1\mu_1 + \bar{\rho}_2\mu_2]},\\
\bar{y} &= \frac{(\lambda_1 + \lambda_2)[\bar{\rho}_1\mu_1 + \bar{\rho}_2\mu_2]}{(\gamma - \bar{\mu})[\bar{\rho}_1\theta_1 + \bar{\rho}_2\theta_2] + \gamma[\bar{\rho}_1\mu_1 + \bar{\rho}_2\mu_2]}.
\end{aligned} \tag{4}$$

Of course, when bottlenecks are located elsewhere, the model can be solved using the same method, but simpler.

3.2. The influence of optimistic unchoking

As introduced in Section 2, a leecher uses choking and unchoking throughout the downloading procedure to encourage and maintain cooperation. Choking has no actual influence on a uni-group swarm, but it should not be neglected when multiple groups coexist. Optimistic unchoking affects the performance and fairness of a heterogeneous swarm in two contradictory ways. On the up side, it offers the dynamic toward the Nash equilibrium state proved in [8], where all groups completely share their uploading bandwidths. On the down side, such tentative uploading is not always rewarded properly. Peers that benefit from this mechanism throughout downloading are those that upload the least, such as free-riders.

With optimistic unchoking in mind, we calculate a leecher's average downloading rates at steady state and the convergence time toward it. Steady state is reached when a peer and any of its regular unchoking objects provide identical uploading rate to each other. In a two-group swarm, it is assumed new leechers only know the existence of the two groups, G_1 with a higher uploading bandwidth and G_2 with a lower one. Leechers also believe each group has over n_u (number of regular unchoking objects) members so they take the Nash equilibrium strategy. In addi-

tion, a peer distributes its uploading bandwidth equally among downloaders.

3.2.1. Converging time

Now we answer a question: how long does it take a peer to reach steady state? Obviously, for a swarm of n groups, only $n-1$ groups need to be considered. Hence, only G_1 peers are discussed in the two-group case.

The regular unchoking list of a G_1 peer stabilizes right after it has identified n_u G_1 neighbors. Since a 30-s optimistic unchoking round is long enough for both sides to learn about each other's uploading bandwidths, we measure how many such rounds are needed. To facilitate a uniform discussion, the bidirectional procedure of optimistic unchoking is equated with a double-paced, locally initiated one. The probability of a G_1 peer finding n_u other G_1 peers right after the kth round is:

$$\begin{aligned}
&\mathbb{P}_{n_u}\{R=k\}\\
&=\mathbb{P}_{n_u-1}\{R\leqslant k-1\}\times P\{G_1 \text{ leecher in round } k\}\\
&=\frac{\binom{N_1-1}{n_u-1}\binom{N_2}{k-n_u}}{\binom{N_1+N_2-1}{k-1}}\frac{N_1-n_u}{N_1+N_2-k}\\
&=\frac{\binom{N_1-1}{n_u-1}\binom{N-N_1}{k-n_u}}{\binom{N-1}{k-1}}\frac{N_1-n_u}{N-k}\\
&=\frac{\binom{N_1-1}{n_u}\binom{N-N_1}{k-n_u}}{\binom{N-1}{k}}\frac{n_u}{k}\\
&=f(n_u;N-1,N_1-1,k)\frac{n_u}{k}. \qquad (5)
\end{aligned}$$

Function f follows hypergeometric distribution. The expectation rounds to reach a steady state can be calculated by:

$$\begin{aligned}
E[R]&=n_u\sum_{k=n_u}^{N-N_1+n_u}f(n_u;N-1,N_1-1,k)\\
&=n_u\binom{N_1-1}{n_u}\sum_{k=n_u}^{N-N_1+n_u}\frac{\binom{N-N_1}{k-n_u}}{\binom{N-1}{k}}. \qquad (6)
\end{aligned}$$

For summation index k, a simple expression of $E[R]$ is not explicitly available. Instead, we plot the relationship between $E[R]$ and G_1's portion among leechers in Fig. 1.

First, it is noticed in Fig. 1 that the number of neighbors or swarm population has almost no influence on the expectation, but more neighbors help reduce G_1 peers' risk of being trapped by G_2 for long. Second, both the expectation and variance of R drop quickly as G_1's portion increases. If G_1 leechers are not rare (say, below 10% in the swarm), they may expect to find enough sound neighbors reasonably soon, probably less than 15 min. Third, the convergence time increases no faster than n_u. Fig. 2 says the difference in expectation between $n_u=3$ and $n_u=4$ is roughly 30%, and the difference in standard deviation is less than 20%.

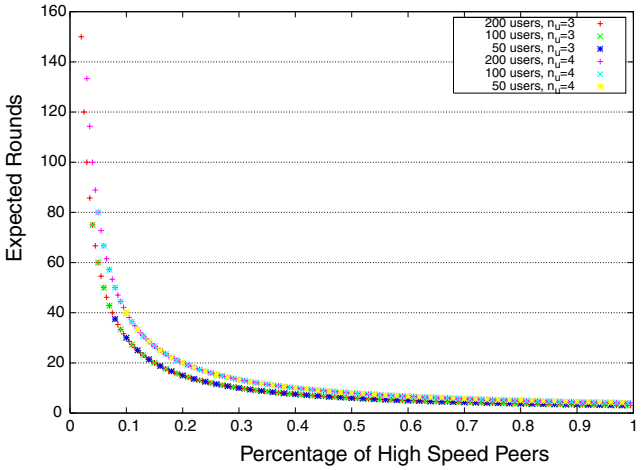

Fig. 1. Convergence time toward steady state.

The above equations and figures explain the *warming up* period often reported by BitTorrent users. Yet most times it is transient. Afterwards, peers will find their average downloading speed more or less stable in the long run.

3.2.2. Average downloading rates

Now we discuss a peer's downloading rate from other leechers in steady state. Even in this state, the downloading rates of a peer may fluctuate because of the random optimistic unchoking. Therefore, peers' expectations of the average value are calculated. Let $\|G_1\|=N_1$, $\|G_2\|=N_2$, $N_1+N_2=N$. The average individual downloading rates of G_1 and G_2, υ_1 and υ_2 are:

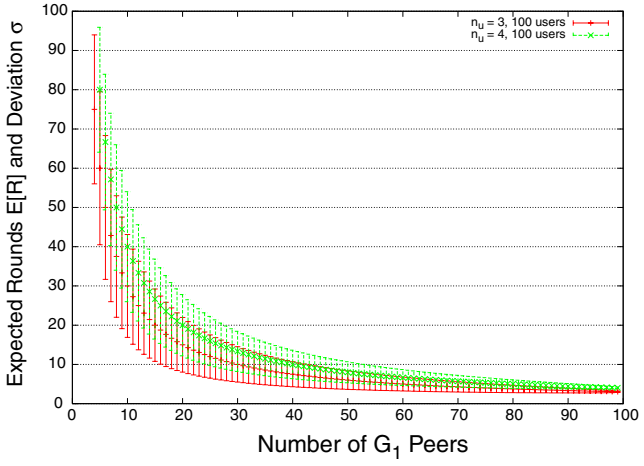

Fig. 2. Expectation and standard deviation of convergence time.

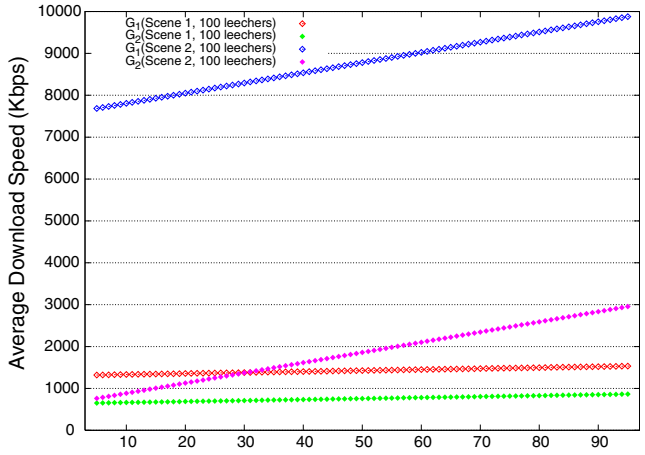

Fig. 3. Average peer downloading rates under different scenarios.

$$\begin{aligned} v_1 &= \frac{n_u}{n_u+1}\mu_1 + \frac{1}{n_u+1}\frac{(N_1-n_u-1)\mu_1+N_2\mu_2}{N_1-n_u-1+N_2} \\ &= \mu_1 - \frac{\mu_1-\mu_2}{n_u+1}\frac{N_2}{N-n_u-1} \\ v_2 &= \frac{1}{N_2}\left(\mu_1 N_2 + \frac{\mu_1-\mu_2}{n_u+1}\frac{N_1N_2}{N_1+N_2-n_u-1}\right) \\ &= \mu_2 + \frac{\mu_1-\mu_2}{n_u+1}\frac{N_1}{N-n_u-1}. \end{aligned} \tag{7}$$

Eq. (7) tells us the downloading rate a G_1 peer expects from neighboring leechers is lower than its uploading bandwidth. On the contrary, a G_2 leecher will get some extra gain. The absolute difference between the average downloading and uploading rate changes linearly according to G_1's percentage in leechers, and the slope is decided by the bandwidth gap between the two groups. Fig. 3 exemplifies the downloading rate of each group under some typical uploading bandwidth combinations. Scene 1 includes 100 leechers made up of T1 (1544 kbps) and ADSL (640 kbps) users, and Scene 2 comprises LAN (10 Mbps) and ADSL (640 kbps) users.

$G1$ peers undergo at most a $\frac{1}{n_u+1}$ discount (compared to its uploading bandwidth) in downloading rate. Nevertheless, the relative gain of a G_2 peer varies greatly. When the bandwidth gap is large, as in Scene 2, G_2 peers' benefit from optimistic unchoking may outweigh their reward from regular unchoking, making downloading bandwidth saturation possible merely by being optimistic-unchoked. In practice, with the continual joining and leaving of leechers, the gain of G_2 users could be more than what has been calculated.

3.3. A general fluid model

To count in optimistic unchoking, we revise the steady-state solution of the fluid model in Section 3.1. In the steady state a peer always download at a stable rate as in Section 3.2.2. Section 3.2.1 demonstrates that the warming up period is relatively short compared with the downloading time, making steady-state statistics a good estimation of the overall situation. In addition, the effect of optimistic unchoking at startup is hard to measure because peers' downloading rates keep changing.

First, the steady-state solution of Eq. (2) is revisited, where uploading bandwidths of the two groups are performance bottlenecks. Each group's downloading rate from other leechers, i.e., μ_i, is replaced by $v_i(\bar{x}_i)$ from Eq. (7). The rate that group i leechers become seeders now becomes $v_i\eta\bar{\rho}_i\bar{x} + \bar{\mu}\bar{\rho}_1\bar{y}, i = 1,2$. And the constraints on system performance bottleneck should be $v_i\eta\bar{x}_i + \bar{\mu}\bar{\rho}_i\bar{y} < C_i\bar{x}_i$, $i = 1,2$. The equations when both G_1's and G_2's downloading bandwidths are saturated are rather simple. When only G_2 peers' downloading bandwidths are saturated, the steady-state equations should be:

$$\begin{aligned} 0 &= \lambda_1 - \theta_1\bar{\rho}\bar{x} - (v_1\eta\bar{\rho}_1\bar{x} + \bar{\mu}\bar{\rho}_1\bar{y}), \\ 0 &= \lambda_2 - \theta_2\bar{\rho}_2\bar{x} - c_2\bar{\rho}_2\bar{x}, \\ 0 &= v_1\eta\bar{\rho}_1\bar{x} + c_2\bar{\rho}_2\bar{x} - \gamma\bar{y}, \\ \bar{\mu}&[\mu_1\eta\bar{\rho}_1\bar{x} + c_2\bar{\rho}_2\bar{x}] \\ &= \mu_1(v_1\eta\bar{\rho}_1\bar{x} + \bar{\mu}\bar{\rho}_1\bar{y}) + \mu_2 c_2\bar{\rho}_2\bar{x}. \end{aligned} \tag{8}$$

This general fluid model simplified the real BitTorrent application in several ways. Being a static model, we ignored the fluctuation of swarm population and downloading speed. At the same time, fluid model does not predict the behavior of extremely small groups very well, because the group behavior may not satisfy the continuous assumptions. Besides, by introducing converging period, to provide a set of deterministic equations about group evolution proves difficult, although numerical analysis can always be applied.

Due to the complexity of the equations, even when there are only two groups, how steady-state performance depends on parameters μ_i, c_i, θ_i, γ remains unclear. Hence, we resort to numerical methods in Section 4 to collect statistics illustrating the performance and fairness of the steady state.

4. Numerical results and analysis

The rich numerical data obtained by solving steady-state models illustrate how a two-group BitTorrent session is influenced by system parameters, and when occasions occur that free-riding becomes appealing. We further distill several keys to the performance and fairness of BitTorrent. Because of scalability and control problems, it is infeasible to conduct so many P2P experiments within a reasonable amount of time. Therefore, we validate our calculations with a handful of available real-life measurements and simulations.

Some baselines are introduced with changes to one or a few parameters when applying numerical analysis on Eq. (1). The trend of average downloading time and group sizes is visualized in Fig. 4. The following baselines are picked referring to the real-life statistics in [4]: $\lambda_1 = 0.002$, $\lambda_2 = 0.012$, $\theta_1 = \theta_2 = 0.0001$, $\gamma = 0.00025$ and $\eta = 1$. Here, we adopt two different settings in Table 1 to reflect different

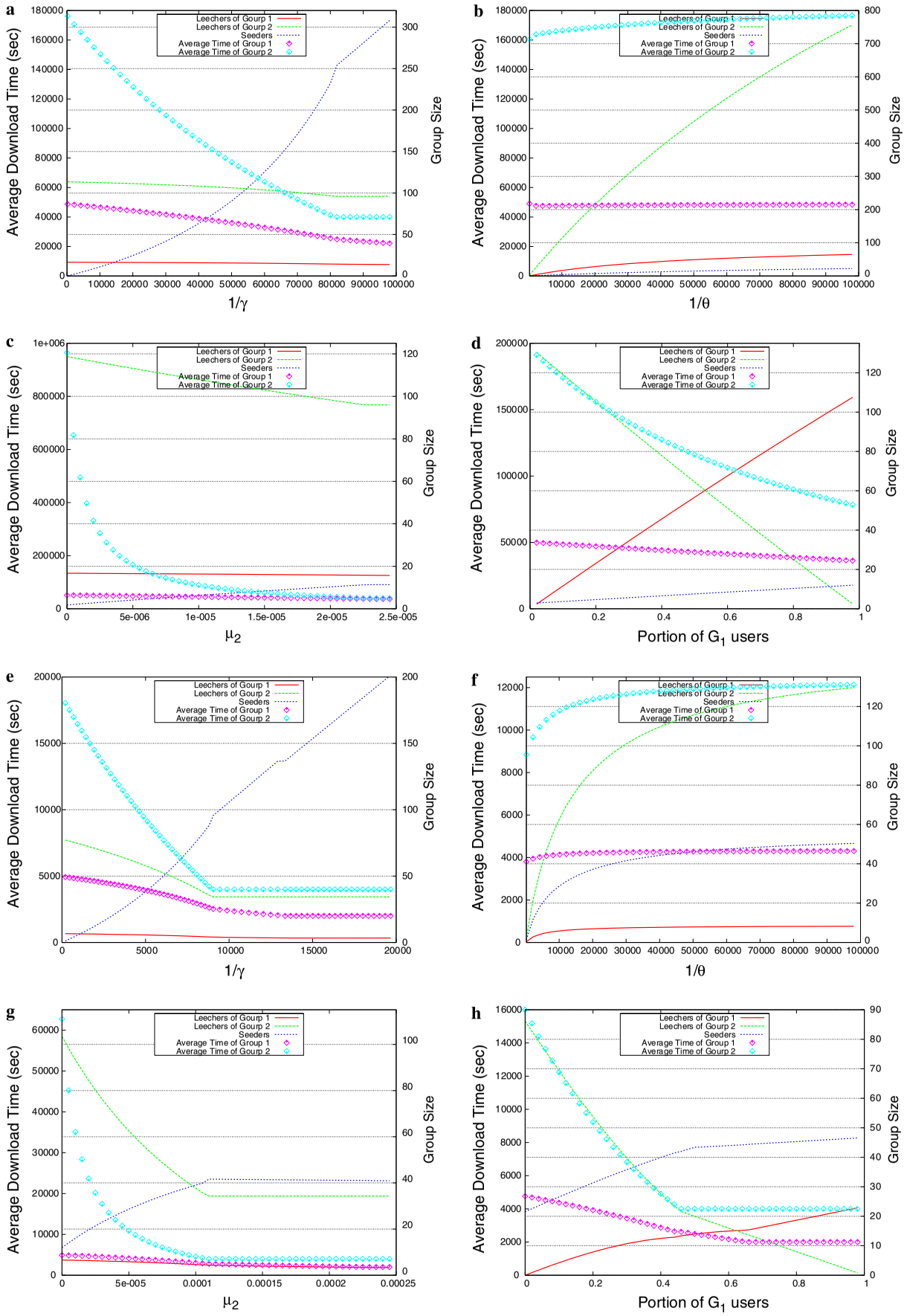

Fig. 4. Change of download speed and group size against different parameters.

Table 1
Default values of bandwidth parameters

Parameter	μ_1	c_1	μ_2	c_2
Setting 1	0.000025	0.00005	0.000005	0.000025
Setting 2	0.00025	0.0005	0.00005	0.00025

file sizes. It is noticed that a G_1 peer has five times the uploading bandwidth of a G_2 peer, but only twice in downloading. It agrees with realistic scenarios where higher-end users usually have more symmetric bandwidths. In Fig. 4, subfigures on the first line adopt setting 1, while the rest adopt setting 2.

4.1. Performance analysis

The maintenance of a session largely depends on seeder population. Sometimes, abundant leechers with few seeders can also survive. So swarm population is the key for a session to survive. Calculation other than in Fig. 4 proves both leecher and seeder populations grow almost proportional to the arriving rate. On the other side, group sizes are negatively correlated to the abandoning rate and G_2 grows much faster than G_1 as this rate drops. Figs. 4(a) and (e) show that the average seeder number quickly approaches 0 when seeders' staying time decreases. But a higher μ_2 or higher percentage of G_1 peers makes the session more survivable. File size plays an important role. Very large files such as in Figs. 4(a)–(d) tend to keep more leechers but less seeders than smaller files. Of the 12 sessions in [5], six sessions targeting larger files (from 580 to 2600 MB) have an average seeder/leecher ratio of 1.00, while the other six targeting smaller files (from 6 to 430 MB) have an average seeder/leecher ratio of 6.75. These statistics demonstrate smaller files typically correspond to higher seeder/leecher ratio.

Some interesting features have been discovered about the efficiency, i.e., average downloading time of each group. Agreeing with the conclusion in a uni-group setting, G_1 users are insensitive to the abandoning rate. However, it is not the case with G_2 peers. In setting 2, a near 40% delay is observed when most G_2 users stick around rather than give up easily. That means the many unfinished downloads (as reported by [4]) actually promote the performance of remaining peers, an effect more obvious on smaller files. The reason is that a higher abandoning rate leads to more G_1 peers within the swarm. Generally, G_2 peers rely on other groups' uploading more than G_1 peers, and hence are more responsive to the parameter changes. On the other hand, the number of seeders is significant to the efficiency of all. For example, with seeders staying longer, average downloading time could be shortened as long as the bandwidth limits permit. In setting 2, when the seeders' average staying time increases from 1000 to 10,000 s, downloading time of a G_1 and G_2 leecher shortens by 50% and 76%, respectively. The evolution concerning μ_2 is worth some special attention here, because Figs. 4(c) and (g) show that allowing a slightly more than zero uploading greatly reduces the downloading time for G_2 peers. This means extremely grudging free-riders will be severely punished, but holding back moderate bandwidths may still lead to acceptable downloading rates. We will return to this phenomenon in Section 4.2.

To find out the scalability of BitTorrent, several other settings have been computed by multiplying all λ_i by a factor. BitTorrent scales well except under some extreme conditions. For large swarms, bandwidth mismatch in regular unchoking rarely occurs. However, the choking algorithm says if the G_1 population is less than n_u, a G_1 peer has to regular-unchoke one or more G_2 peers and suffer a lower downloading rate. In this sense, big swarms, or swarms with high arriving rates and low leaving rates, have stabler and more predictable performance.

4.2. Analysis of the fairness problem

The fairness problem becomes a concern of BitTorrent on certain occasions. Taking a close look at μ_2, we find in Fig. 4(c), a G_2 leecher' providing 60% the uploading bandwidth of G_1 actually downloads at 62% the rate of G_1. Under setting 2, a G_2 peer only needs to provide 40% G_1's uploading bandwidth to receive a 59% relative downloading rate. Table 2 lists the thresholds at which peers' downloading bandwidths become saturated and the corresponding group may readily restrict uploading without harming their downloading. In both settings, G_2 peers reach saturation before G_1. Besides, it is more likely to achieve full-speed download with smaller files, mainly because of the abundant seeders.

As leechers keep abandoning the session, not all peers will finally get a complete file. The probabilities to finish download under varying circumstances are summarized in Fig. 5. Assuming a constant abandoning rate in six of the eight subfigures presented in Fig. 5, we expect their finishing probabilities to be negatively correlated to the downloading time. So it is not surprising that G_1 leechers always own better chances to finish download. Since there exist lower bounds of downloading time, finishing probabilities have upper bounds less than 1. Targeting a very large file, Figs. 5(a)–(d) show the majority of the population fails to obtain it at last. Obviously seen from Figs. 5(e)–(h), smaller files provide better opportunities in this regard. Under default setting 2, G_1 and G_2 peers have probabilities of 0.672 and 0.181 to complete download. The overall proba-

Table 2
Thresholds of full-capacity downloading

Parameters	Setting 1		Setting 2	
	G_1	G_2	G_1	G_2
γ	–	<1.22e−5	<7.49e−5	<1.11e−4
μ_2	–	>2.26e−5	>2.31e−4	>1.08e−4
G_1 (%)	–	–	>65.5	>45.2

"–" means such threshold does not exist.

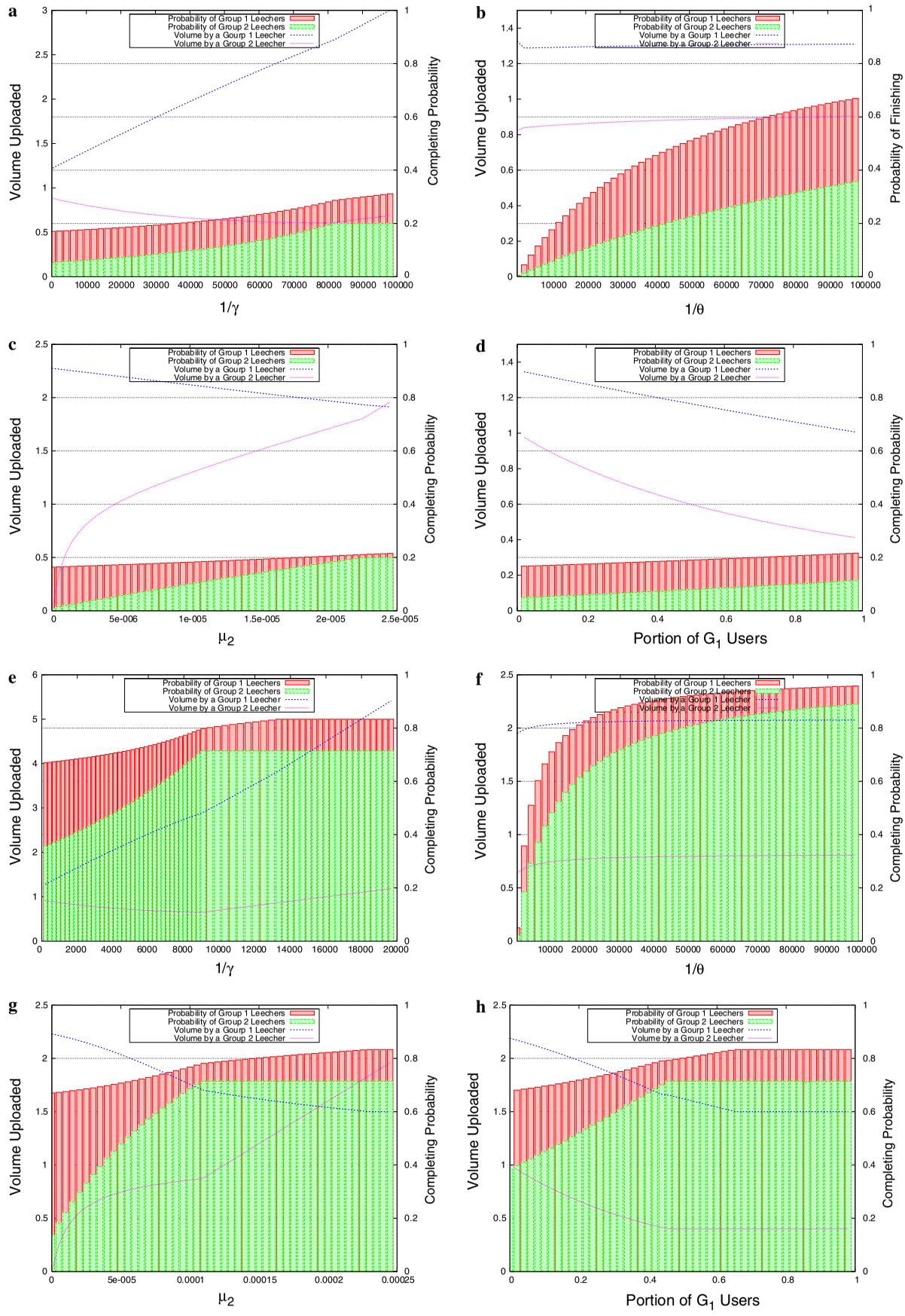

Fig. 5. Completing probability and data uploaded by completing peers.

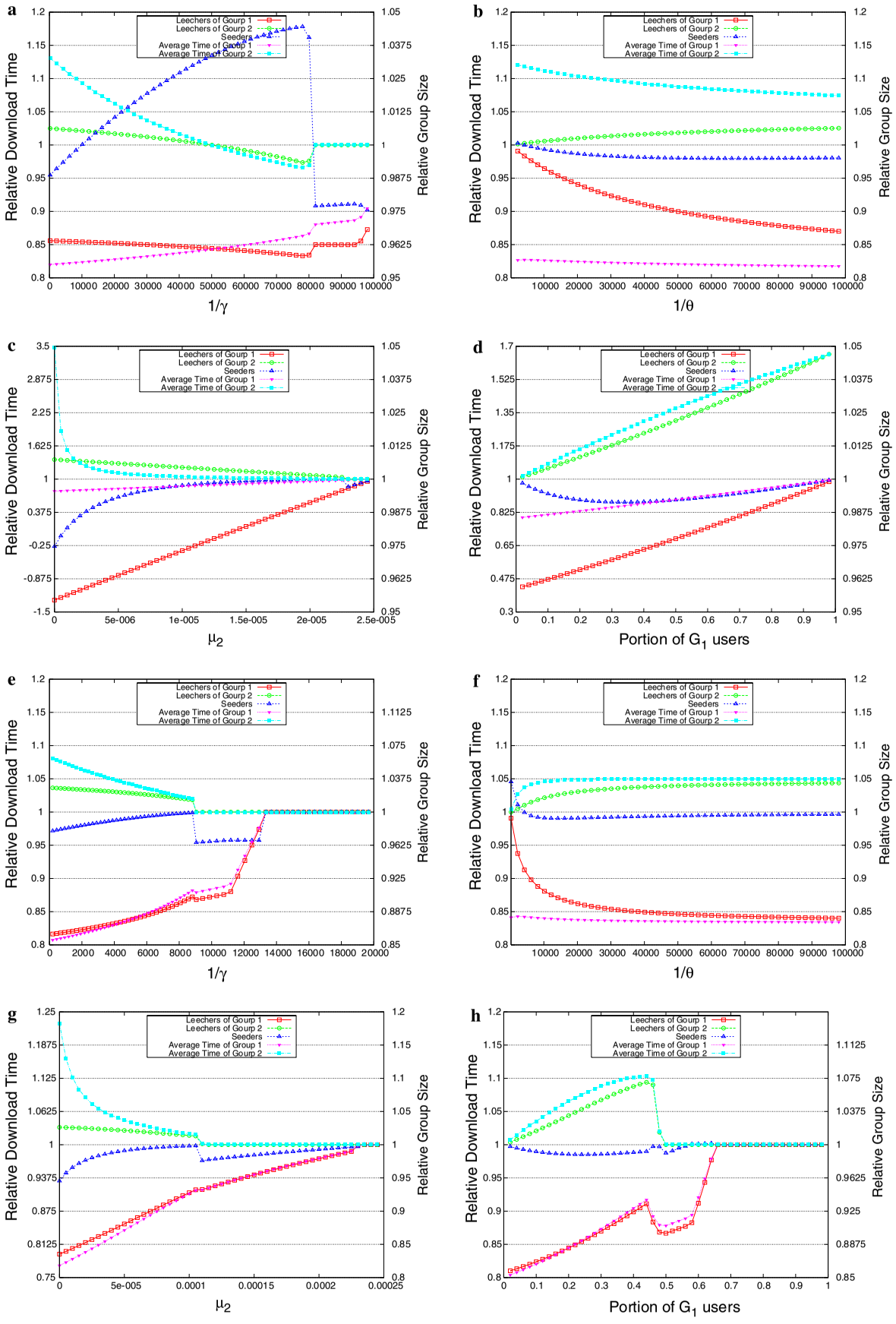

Fig. 6. Effect of eliminating optimistic unchoking.

bility of 27% is very close to the real-world observation of 24% in [4].

How much data are uploaded by peers getting a complete file is also calculated and displayed in Fig. 5. Under default bandwidth settings, G_1 peers contribute more than one copy on average, however, other parameters change, while G_2 peers always upload less than one copy. Only increasing G_2's uploading capacity, i.e., μ_2, can induce more data served by G_2. With a smaller file size, G_1 peers leave the session with a higher contribution rate (against downloading size). However, data contributed by G_2 peers are not much affected by file size.

The combination of optimistic unchoking and seeders' unselfish uploading makes free-riding more attractive. Under default setting 1 (setting 2), G_1 leechers take 21.7% (10.9%) more time in the general fluid model compared with those in a model without inter-group data exchange. From G_2 peers' view point, with optimistic unchoking they save 10.3% (4.4%) downloading time. Meanwhile, without optimistic unchoking the bandwidth saturation becomes less likely for G_2 but easier for G_1. Based on a similar observation in simulation, a bandwidth estimation method is suggested in [6] to replace optimistic unchoking.

By collecting statistics of sessions without optimistic unchoking but alternative peer-picking methods like bandwidth estimation, we compare their relative group sizes and downloading time against sessions at the beginning of this section in Fig. 6. Eliminating tentative uploading leads to less G_1 leechers but usually more G_2 leechers within the swarm. It also reduces the seeder population in general. Exceptions occur in Fig. 6(a) where the relative seeder population reaches a maximum of 1.045, or in Fig. 6(f) when θ is very large. Relative populations of peer groups often encounter some drastic changes, because different thresholds of bandwidth saturation exist for inter-group data exchange enabled and disabled sessions. G_1 leechers expect a shorter downloading time without the effect of optimistic unchoking. G_2 leechers, though undergo a degraded performance most of the time, could even receive better uploading rate from other peers as in Fig. 6(a). We might recall that optimistic unchoking has been replaced to alleviate the fairness problem. Then, this arresting phenomenon indicates getting rid of optimistic unchoking could, in some case at least, induce more unfairness.

4.3. Discussion

(i) *Security*. After demonstrating BitTorrent is quite scalable facing flash arrivals of normal peer composition, it remains doubtful if it can resist abundant malicious free-riders trying to deter normal downloading. Our convergence time of choking algorithm in Section 3.2.1 shows with very low percentage of high-bandwidth peers, they may be easily trapped by low-bandwidth users and find it hard to achieve their ideal downloading rates. We note that this potential risk will not be remedied by optimistic unchoking alternatives, like bandwidth estimation. The fundamental problem is neighbors are selected by chance rather than by credit and trust. What's more, once neighbors are chosen, the local peer does not actively close connections based on performance, and therefore has little room for new neighbors.

(ii) *Seeder*. The significance of seeders is often underestimated in theoretical studies. A long-staying peer can upload much more data as a seeder than in leecher state, and the existence of seeders is crucial to maintain a session. What's more, with plenty of seeders, a session with a low utilization rate of uploading bandwidths can still be very efficient. Seeders will also be the only hope of low uploading bandwidth leechers if optimistic unchoking is replaced by peer-picking algorithms free of data exchange. Unfortunately, the current BitTorrent protocol lacks incentives to keep seeders around, other than not providing an automatic way to exit a session. Being file-oriented and depending much on seeders, performance of BitTorrent sessions can be quite unstable even with the same composition of peers. Moreover, we notice a distinct relationship between file size and seeder population, making sharing smaller files more efficient than larger ones. Nevertheless, if the seeder leaving rate rises, sessions of smaller files tend to cease more easily than larger ones.

(iii) *Free-riding*. BitTorrent is vulnerable to free-riding under certain circumstances. Previously, experimental approaches often believed that BitTorrent induces free-riding, but there lacks theoretical support for this phenomenon. Fig. 4 and Table 2 provide sufficiency conditions of free-riding in a theoretical model. Here, the question is, is it really necessary to force low-bandwidth peers sharing as much data as their high-bandwidth neighbors? Optimistic unchoking alternatives do reduce, sometimes but not always, the likelihood of free-riding among G_2 leechers, on the contrary, they make free-riding more appealing to G_1 leechers. Luckily, in BitTorrent the system performance is always positively related to the free-riding incentives. In other words, better fairness could be due to performance deficiency. Therefore, the fairness problem of a normal BitTorrent session, i.e., without a flash-crowd of malicious free-riders, is not critical.

5. Conclusion

In this paper, we have studied BitTorrent-like networks using a general fluid model. The emphasis is laid on the performance and fairness of the steady-state performance. With numerical methods, the influences of major system parameters as well as the optimistic unchoking mechanism are illustrated and discussed over a large number of parameter settings. From our results, we find seeders play a key roll in achieving good overall performance, yet there lacks respective encouraging mechanisms. We also find BitTorrent is not strictly scalable or safe against abundant maliciously selfish leechers. The free-riding incentives in

BitTorrent are stronger than previously believed in theoretical study, but they are not devastating to the overall performance.

Our future work includes developing a modeling toolkit based on our current model, which can calculate steady-state statistics and illustrate the whole procedure of swarm evolution. Also, as there has been statistics about swarms sharing different sizes of file, we will perform more experiments to better reveal the relationship between file size and swarm characteristics.

References

[1] S. Sen, J. Wang, Analyzing peer-to-peer traffic across large networks, IEEE/ACM Transactions on Networking 12 (2) (2004) 219–232.
[2] URL <http://www.bittorrent.com/>.
[3] T. Mennecke, Bittorrent remains powerhouse network (January 2005). URL <http://www.slyck.com/news.php?story=649/>.
[4] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. Felber, A.A. Hamra, L. Garcs-Erice, Dissecting bittorrent: five months in a torrents lifetime, in: Proc. PAM 2004, 2004, pp. 1–11.
[5] P.M. Arnaud Legout, Guillaume Urvoy-Keller, Understanding bittorrent: an experimental perspective, Tech. Rep. inria-00000156, version 3, INRIA Sophia Antipolis & Institut Eurecom (November 2005).
[6] A.R. Bharambe, C. Herley, V.N. Padmanabhan, Analyzing and improving bittorrent performance, Tech. Rep. MSR-TR-2005-03, Carnegie Mellon University and Microsoft Research (February 2005).
[7] X. Yang, G. de Veciana, Service capacity of peer to peer networks, in: INFOCOM '04, 2004, pp. 2242–2252.
[8] D. Qiu, R. Srikant, Modeling and performance analysis of bittorrent-like peer-to-peer networks, in: SIGCOMM '04, 2004, pp. 367–378.

Yao Yue is an ME student in the Department of Computer Science at Tsinghua University, China. Her research interests include peer-to-peer networks, network processors, performance evaluation, and network measurement. Yue has received her BS in physics and mathematics from Tsinghua University.

Chuang Lin is a professor and the head of the Department of Computer Science and Technology, Tsinghua University. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. Lin has a PhD in computer science from Tsinghua University. He serves as the general chair for ACM SIGCOMM's 2005 Asia workshop, the associate editor of the IEEE Transactions on Vehicular Technology and the Area Editor of Journal of Parallel and Distributed Computing. He is a senior member of the IEEE.

Zhangxi Tan is an PhD student in the Department of Computer Science at University of California, Berkeley. His research interests include computer architecture, performance evaluation, and resource management in computer networks. Tan has a BE in electronic engineering and a ME in computer science, both from Tsinghua University.