Analyzing Multiple File Downloading in BitTorrent

Ye Tian, Di Wu and Kam-Wing Ng
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {ytian, dwu, kwng}@cse.cuhk.edu.hk

Abstract

Previous studies show that more than 85% of the peers have joined multiple torrents in BitTorrent, but theoretical work on multiple files BitTorrent downloading is rare. In this paper, we first consider the scenario of multi-torrent downloading. We present a fluid-model based analysis on the multi-torrent concurrent downloading scheme, which is implicitly adopted in practical applications, and quantitatively compare its performance with an alternative scheme of multi-torrent sequential downloading. We also consider the scenario of multi-file torrent downloading (e.g. multiple files shared within a single torrent), and find that the scheme of multi-file torrent concurrent downloading, which is explicitly engaged in practical applications, is inefficient. A new scheme named collaborative multi-file torrent sequential downloading is proposed, and we show via numerical analysis that the download performance could be improved by collaboration among the peers in different subtorrents. Finally, we propose a self-adaptive mechanism for practically deploying our multi-file torrent downloading scheme in a distributed fashion under situations when correlation among the files and majority peers' behaviors are unknown.

1 Introduction

BitTorrent [1] is a peer-to-peer (P2P) file sharing protocol which is extremely successful in recent years. The main feature of BitTorrent, as well as other similar protocols, is that a file is divided into many chunks, and for each peer, it will download chunks from other peers while uploading the chunks it keeps at the same time. For the partner selection, a tit-for-tat (TFT) strategy is adopted, in which a peer simply chooses to upload to those peers from which it could download at the largest rates, and it will follow the local rarest first strategy (LRF) to choose to download the chunks which are rarest replicated among the neighboring peers. For those peers which have finished downloading all the chunks, they will upload them in an altruistic fashion. Usually in a torrent, people call the peers which are currently downloading the content as the downloaders and the peers which have already finished the job but have not quit the torrent as the seeds.

Due to its success in real world, many studies have been proposed to understand the BitTorrent's performance in recent years [2] [3] [4] [5] [6] [7] [8] [9], these studies provide insightful and accurate results on the system's performance. However, most of these researches are single-file-single-torrent based, assuming that the user enters into a torrent for exactly one file with all the available bandwidth, and there is no correlation between the files requested by the users. While in practice, many of the files published via BitTorrent are interest-correlated, which means users who have requested one of them will be very likely to request another. Examples of the interest-correlated files include the different versions of the computer games under the same theme, the movies featuring the same superstar, or episodes of a TV play.

In this paper, we consider different scenarios of interest-correlated multiple file downloading in BitTorrent. Concretely, two scenarios are under consideration: First, we study the multi-torrent downloading scenario, in which the user enters into separate torrents for different files. Currently, most of BitTorrent client softwares will download the files concurrently, and each instance of a BitTorrent download process competes the available bandwidth with others. Note that the files requested in separate torrents may have weak or strong correlation among themselves. So our question is: taking the correlation among the files into consideration, is this multi-torrent concurrent downloading (MTCD) scheme efficient? The second scenario under consideration is the multi-file torrent downloading. In this scenario, several files are published within a single torrent by the publisher, and the users could choose to request a subset of the files or download them all. Currently, most BitTorrent client softwares which support this functionality just download the chunks of the requested files randomly, as if it is downloading the chunks of a single file, in this way, the files are actually downloaded in a concurrent fashion, and we call it the multi-file torrent concurrent downloading (MFCD) scheme. For MFCD, our question is: is it efficient? Since files published within a single torrent is usually highly interest-correlated, another question is: is it possible to improve the downloading efficiency by making use of the high correlation among the files, and among the peers downloading them.

We present a fluid model based analysis on the multi-torrent downloading scenario, and find that the MTCD scheme, which is implicitly engaged by the users, are not

efficient enough, compared with the alternative scheme of multi-torrent sequential downloading (MTSD). This inefficiency is more obvious when the files published are highly interest-correlated. For the multi-file torrent downloading scenario, our analysis shows that the performance of the MFCD scheme, which is explicitly engaged by the content publishers and the BitTorrent client software, could be improved by exploiting the correlation among the peers requesting for the same set of files in the single torrent. Motivated by the observation, we propose a novel scheme called collaborative multi-file torrent sequential downloading (CMFSD) in this paper. In our scheme, the content publishers also publish the highly interest-correlated files in a single torrent, but the peers download them in a collaborative way by serving as partial seeds. We show via fluid model based analysis that the performance gets improved intensively for high interest-correlated files. Finally, a self-adaptive mechanism is proposed for practical deployment of the protocol by the individual peer.

1.1 Related Work

For such a widely deployed protocol as BitTorrent, real-world application measurement [2] [3] [4] is the most direct way for people to understand the system's performance. In [2], a five-month analysis on the tracker's log file of a BitTorrent system is presented, the author shows that BitTorrent is realistic and inexpensive, compared with traditional protocols such as ftp and http. A measurement [3] on the BitTorrent/Suprnova architecture is given recently, the influence on the metrics as content's availability and integrity by the website user's behavior are systematically measured and studied. Also in a recent measurement [4], the new peer arrival is discovered to be deceasing exponentially, and the universal phenomena of peers joining multiple torrents is observed. On the other hand, to have an insightful understanding of the performance of BitTorrent, some theoretical analysis are proposed. A branching process model is discussed for studying the transient regime of the BitTorrent system, and a Markov chain model is proposed for capturing the system's service capacity under the stable state in [6]. A fluid model is presented in [7], where the expressions of the numbers of the downloaders and the seeds could be obtained under the assumption that peers' arrival and departure follow a Poisson process. The analysis also shows that BitTorrent achieves very good scalability. However, recent studies [8] [9] reveal systematic inefficiencies at the beginning and the end of the peer's download job progess in BitTorrent. Enhancements to BitTorrent could be found in Slurpie [10] and Avalanche [11]. The Slurpie system integrates techniques such as group size estimation, back off and bandwidth estimation to improve the utilization of the link bandwidth and to decrease the burden of the topology server. For Avalanche, the pieces of the file are encoded both on the source and on the nodes, which makes the block propagation of the file more efficient than transmitting the uncoded blocks.

Our unique contribution is this paper is that unlike the previous studies, which focus only on one-file-one-torrent system, we work on the performance and possible improvement of a multiple file system with correlations among the files. To our best knowledge, similar topics are only studied in [4], however, in [4], the authors make efforts to prolong the torrent's lifetime, but our work aims to improve individual user's performance.

The rest of the paper is organized as follow: we discuss the fluid model in Section 2; and in Section 3, performance analysis based on the fluid model for different downloading schemes is developed, we also present our proposal and analysis on the collaborative multi-file torrent sequential downloading scheme in this section; numerical evaluations and discussions on deployment are given in Section 4 and finally we conclude this paper in Section 5.

2 The Fluid Model for BitTorrent

Models of fluid-based methodologies have been proved powerful for analyzing the performance of the large scaled networking systems. In [13], a stochastic fluid model is used for studying the performance of the P2P web catch system – Squirrel [12]; and in [7], a fluid model is developed for evaluating the BitTorrent system. Although these models are simple, they provide accurate and insightful results for the performance issues, such as the hit probability of Squirrel and the downloading time for BitTorrent.

We develop our analysis based on the model of [7]. In the model, $x(t)$ and $y(t)$ are used to represent the numbers of the downloaders and the seeds at time t respectively, and two ordinary differential equations (ODE) are developed to study the population evolution for these two different kinds of peers as follows:

$$\begin{cases} \frac{dx(t)}{dt} = \lambda - \mu(\eta x(t) + y(t)) \\ \frac{dy(t)}{dt} = \mu(\eta x(t) + y(t)) - \gamma y(t) \end{cases}$$

Here we just assume that the available download bandwidth of a peer is much larger than the available upload bandwidth, so the system's performance is constrained by the upload capability of the peers. The meanings of the parameters in the model are listed in Table 1.

$x(t)$	num. of the downloader peers in the torrent at time t
$y(t)$	num. of the seeds in the torrent at time t
λ	entry rate of new peers
η	file sharing efficiency between two downloader peers
μ	upload bandwidth
γ	rate of the seeds departing the torrent

Table 1. Parameters in BitTorrent fluid model

The parameter η is modeled as the ratio of a downloader's service capability in uploading the chunks compared with the capability of a seed. In [7], the authors regard that η is the probability that a downloader could upload chunks to a connected peer, and prove that this probability is near 1 when

the number of the chunks is large enough. However, in [2], it is reported that over five months, the seeds have contributed more than twice the amount of the data sent by the downloaders, while the portion of the seeds among the total peer population is consistently lower than the portion of the downloaders after the initial phase of flash crowds (referring to Fig. 2 (a) and (b) of [2]). Based on this observation, we believe that the value of η should be 0.5 at maximum. The relatively low uploading efficiency of the downloaders could be explained with the tit-for-tat strategy played by the downloaders in exchanging the chunks: unlike the altruistic uploading by the seeds, the downloaders only upload chunks conditionally. In this paper, we will simply set a value of 0.5 for η, same as the value chosen in [6].

The fluid model in [7] treats all the downloaders and the seeds uniformly, however, in this paper, we wish to know the performance of the peers categorized into different classes, respecting their upload/download bandwidths. Suppose the peers in a torrent could be categorized into S classes as $\{C_1(\mu_1, c_1), ..., C_S(\mu_S, c_S)\}$, where a peer of class C_i has a upload bandwidth of μ_i and a download bandwidth of c_i, and there are $x_i(t)$ downloaders and $y_i(t)$ seeds of class C_i at time t. We make two assumptions:

- For the downloaders of a class C_i, the total service they received from the downloaders in $\{C_1, ..., C_S\}$ at time t is proportional to their upload bandwidth μ_i and their population $x_i(t)$ as $\frac{x_i(t)\mu_i}{\sum_{l=1}^{S} x_l(t)\mu_l} \sum_{l=1}^{S} \eta x_l(t)\mu_l = \eta x_i(t)\mu_i$.
- For the downloaders of a class C_i, the total service they received from the seeds in $\{C_1, ..., C_S\}$ at time t is proportional to their download bandwidth c_i and their population $x_i(t)$ as $\frac{x_i(t)c_i}{\sum_{l=1}^{S} x_l(t)c_l} \sum_{l=1}^{S} y_l(t)\mu_l$.

The first assumption is based on the TFT strategy in chunks exchanging between the downloaders; and for the second assumption, it is an approximation of the altruistic behavior of the seeds, since they just upload to peers according to their ability in receiving the chunks.

3 Performance Analysis for Multiple file Downloading System

3.1 The Server-Torrent Architecture

BitTorrent is a P2P file sharing protocol which does not rely on a centralized placement of the shared content. However, in the real world deployment, at least two centralized components are required: a web server and a tracker server. The web server provides an indexing service: it just provides a collection of links to the metadata file of a BitTorrent shared content (with .torrent as the extension name). Users could obtain the metadata of the existing torrents, and probably the number of the downloaders and the seeds currently in the torrent from the server. Examples of the BitTorrent indexing web server include and BitTorrent@China [14] and

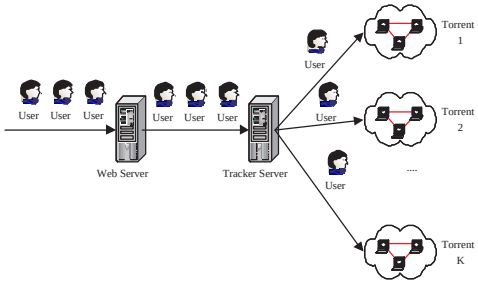


Figure 1. The server-torrent architecture.

TorrentSpy [15]. The web server could also be in the form of a forum, in which the publishers upload the metadata as an attachment. After the user obtains the metadata file, the BitTorrent client software is launched and connects to the tracker server, and the user enters into the corresponding torrent to download the content.

Usually an indexing web server may provide links to torrents belonging to many tracker servers, and conversely a tracker server's torrent metadata file might be published on many web servers. For simplicity, in this paper we assume that there is only one web server and one tracker server as shown in Figure 1, the content published on the web server will only be served by the tracker server, and the web server will only publish the content served by the tracker server. In the followings we will investigate the performance of different multiple file downloading schemes under this server-torrent architecture.

3.2 Multi-Torrent Concurrent Downloading

We first consider the situation that users enter into separate torrents to download different files. The scheme of multi-torrent concurrent downloading (MTCD) is studied in this part. In MTCD, a user will enter multiple torrents for the files he/she has interest in and download them concurrently. Suppose there are K files and consequently K torrents in the system, denoted as $\{t_1, t_2, \ldots, t_K\}$. For a user who downloads multiple files simultaneously, multiple peers are generated: one for each torrent. If the available upload bandwidth for a user is μ, then for each peer, the average available bandwidth is μ/i, where i is the number of files the user has concurrently requested. The download bandwidth is also divided into i shares. We further categorize the peers engaged in the K torrents into K classes: for the ith class peer, its user has requested i files simultaneously. For a particular torrent, say t_j, we use λ_j^i for the entry rate of the new peers of the ith class, and use $x_j^i(t)$ and $y_j^i(t)$ to denote the number of the downloaders and the seeds of the ith class in t_j respectively, then the service contribution from the ith class downloaders in torrent t_j could be expressed as $\eta\frac{\mu}{i}x_j^i(t)$; and for the ith class seeds, the service provided by them is $\frac{\mu}{i}y_j^i(t)$. Under the assumptions we have made in Section 2, the downloaders of the ith class receive a portion of $\frac{\frac{1}{i}x_j^i(t)}{\sum_{l=1}^{K}\frac{1}{l}x_j^l(t)}$ for the total service provided by all the downloaders and the seeds. Ag-

gregating all the analysis, we have the following fluid model for torrent t_j as:

$$\begin{cases} \frac{dx_j^i(t)}{dt} = \lambda_j^i - \eta\frac{\mu}{i}x_j^i(t) - \frac{\frac{1}{i}x_j^i(t)}{\sum_{l=1}^{K}\frac{1}{l}x_j^l(t)}\sum_{l=1}^{K}\frac{\mu}{l}y_j^l(t) \\ \frac{dy_j^i(t)}{dt} = \eta\frac{\mu}{i}x_j^i(t) + \frac{\frac{1}{i}x_j^i(t)}{\sum_{l=1}^{K}\frac{1}{l}x_j^l(t)}\sum_{l=1}^{K}\frac{\mu}{l}y_j^l(t) - \gamma y_j^i(t) \end{cases} \tag{1}$$

where $i, j \in \{1, 2, ..., K\}$.

Under the stable state in which $\frac{dx_j^i(t)}{dt} = 0$ and $\frac{dy_j^i(t)}{dt} = 0$, we have

$$\begin{cases} x_j^i = i\lambda_j^i\frac{\gamma\sum_{l=1}^{K}\lambda_j^l - \mu\sum_{l=1}^{K}\frac{1}{l}\lambda_j^l}{\gamma\mu\eta\sum_{l=1}^{K}\lambda_j^l} \\ y_j^i = \frac{\lambda_j^i}{\gamma} \end{cases}$$

for the numbers of the ith class downloaders and the seeds in torrent t_j respectively.

We use the online time as the metrics for performance evaluation, which is the sum of downloading time and the time of the peer serving as a seed. Applying Little's law [16], the average online time for the ith class user at the stable state is:

$$T_i^{MTCD} = \frac{x_j^i}{\lambda_j^i} + \frac{1}{\gamma} = i\frac{\gamma\sum_{l=1}^{K}\lambda_j^l - \mu\sum_{l=1}^{K}\frac{1}{l}\lambda_j^l}{\gamma\mu\eta\sum_{l=1}^{K}\lambda_j^l} + \frac{1}{\gamma} \tag{2}$$

We could see that a user's average download time is proportional to the number of the files he/she has requested.

3.3 Multi-Torrent Sequential Downloading

Another way for a user to obtain multiple files from different torrents is to download them sequentially instead of concurrently. In the multi-torrent sequential downloading scheme (MTSD), since a user enters into only one torrent at a time, all the bandwidth of the user is engaged in one downloading process, thus we could apply the results in [7] directly in our analysis.

Suppose for a particular torrent, say torrent t_j, we use $x_j(t)$ and $y_j(t)$ to denote the numbers of the downloaders and the seeds respectively, and assume that the entry rate of the new peers for t_j is λ_j, then the fluid model for torrent t_j is:

$$\begin{cases} \frac{dx_j(t)}{dt} = \lambda_j - \mu\eta x_j(t) - \mu y_j(t) \\ \frac{dy_j(t)}{dt} = \mu\eta x_j(t) + \mu y_j(t) - \gamma y_j(t) \end{cases} \tag{3}$$

for $j \in \{1, 2, ..., K\}$.

Following the same procedure as in [7] and in the previous section, we have the average download time in a torrent as $T = \frac{\gamma-\mu}{\gamma\mu\eta}$ and the average online time for the ith class peer (peers requesting i files) as

$$T_i^{MTSD} = i(T + \frac{1}{\gamma}) = i(\frac{\gamma-\mu}{\gamma\mu\eta} + \frac{1}{\gamma}) \tag{4}$$

where $\gamma > \mu$. As in the multi-torrent concurrent downloading, the average download time is proportional to the number of files the user has requested.

If the number of torrents is 1, we can see that our analysis results in (2) and (4) degenerate and conform to the result of the single torrent performance in [7] (set $K = 1$, $i = 1$ and remember that the model is constrained by peers' upload bandwidth). In this way, we could prove the correctness of our model.

3.4 Multi-File Torrent Concurrent Downloading

Sometimes, the publisher will publish several files in just one torrent within one metadata file, and many BitTorrent clients allow the user to choose a subset of the files included in a torrent to download. Files published within a single torrent are usually highly interest-correlated (e.g., TV series), and many users are very likely to choose almost all the files in the torrent to download. Currently, most of the BitTorrent client softwares do not differentiate the single file content and the multiple file content, but download the chunks randomly. So, the multiple files within a single torrent are actually downloaded in a concurrent fashion, and we call this downloading scheme as the multi-file torrent concurrent downloading (MFCD).

In MFCD, suppose a peer has chosen i files, we could simply view it as a set of i virtual peers, and each virtual peer is competing bandwidth with the other virtual peers. On average a portion $1/i$ of upload/download bandwidth is assigned to each virtual peer. Under the scheme of MFCD, a torrent is divided into K subtorrents, and a peer could enter multiple subtorrents with multiple virtual peers. In this case, the MFCD scheme could be viewed to be equivalent to the MTCD scheme, and the only exception is that in the MTCD scheme, the peers of a user depart the system independently, while the virtual peers belonging to the same real peer are departing as a whole under MFCD. However, this difference will not influence our application of the previous results on MTCD, since in both cases the average service time of a seed is $1/\gamma$. We will use the result in (2) for evaluating the performance of the MFCD scheme.

Although the MFCD scheme is equivalent to the MTCD scheme in the fluid model, it is worth pointing out that the files being downloaded in MFCD and MTCD have different levels of interest correlation. Usually, users will download all the files provided in a single torrent in the MFCD scheme, since these files are highly interest-correlated, such as TV plays. However, although it is observed in [4] that many peers request more than one files in its lifetime, it is hard to estimate how many users will download the same set of files simultaneously in the MTCD scheme (It depends on the type and the popularity of the content). In this sense, we believe that it is promising to improve the download performance of MFCD by allowing the peers to collaborate in the multi-file torrent downloading scenario, since all the files are highly correlated. Motivated by this consideration, we

present a scheme of collaborative multi-file torrent sequential downloading (CMFSD) in the next section.

3.5 Collaborative Multi-File Torrent Sequential Downloading

In the MFCD scheme studied in the previous section, all the peers are treated uniformly, even if some peers are belonging to the same user. However, when the files shared are highly interest-correlated, it is very likely that a downloader in one subtorrent could be a potential seed in another subtorrent since it has obtained the complete file. In this case, allowing collaboration among subtorrents will intuitively improve the overall performance of the multi-file torrent downloading system. Based on this observation, in this part, we propose a scheme of collaborative multi-file torrent sequential downloading (CMFSD), and present a fluid-model based analysis on its performance.

In CMFSD, K interest-correlated files are published within a single torrent, in which there are K subtorrents, one for each file. A peer entering the torrent could specify the files it will request in the torrent, but could be allowed to download them sequentially, and the downloading sequence is randomized. A peer under CMFSD will allocate all its downloading bandwidth in the subtorrent it is currently in. But for a downloader which has requested multiple files and has finished some of them, it will allocate a part of its upload bandwidth in one of the subtorrent for which it has a complete file, serving as a seed partially in the multi-file torrent downloading system, and it uses the rest of the upload bandwidth to play TFT in the subtorrent in which the corresponding file it is currently downloading. Actually, we could view such a peer as a combination of a virtual downloader and a virtual seed.

We call the peers which have requested i files in the torrent as the ith class peers. For a torrent with K files, suppose the entry rate of the ith class peer is λ_i, and we use $x^{i,j}(t)$ and $y^i(t)$ $(1 < j \leq i)$ to denote the number of the ith class peers which are downloading their jth file (already finished $j-1$ files) and the ith class seeds in the torrent at time t respectively.

For a downloader of $x^{i,j}(t)$ in the system, which means the downloader peer has requested i files and has already finished $j-1$ of them, the upload bandwidth μ is divided into two parts, bandwidth of $\rho\mu(0 \leq \rho \leq 1)$ is allocated for its virtual downloader in the subtorrent in which the peer is downloading its jth file; and the remaining bandwidth $(1-\rho)\mu$ is allocated for its virtual seed serving one of the $j-1$ files it has finished. On the other hand, for a downloader of $x^{i,j}(t)$ in the system, it could receive service from three sources: 1) the other downloaders in the same subtorrent; 2) the virtual seeds of the downloaders which serve in the same subtorrent; 3) the real seeds serving in the subtorrent which have finished all the files requested.

We define a function $P(i,j)$ as: $P(i,j) = 1$, when $i = 1$, or $j = 1$, this is for a peer which is downloading its first file, thus has no completed file to serve as a virtual seed; and $P(i,j) = \rho$, $\rho \in [0,1]$, when $i \neq 1$ and $j \neq 1$, this is for a downloader with at least one completed file. With the function $P(i,j)$, the total service provided by the first source could be expressed as $\sum_{l=1}^{K}\sum_{m=1}^{l}\mu\eta P(l,m)x^{l,m}(t)$, and the total service provided by the second and the third source are $\sum_{l=1}^{K}\sum_{m=1}^{l}\mu(1-P(l,m))x^{l,m}(t)$ and $\sum_{l=1}^{K}\mu y^l(t)$ respectively.

Under the assumption made in Section 2, for the peers in $x^{i,j}(t)$, the total amount of service received from the first source is $\frac{P(i,j)x^{i,j}(t)}{\sum_{l=1}^{K}\sum_{m=1}^{l}P(l,m)x^{l,m}(t)}(\sum_{l=1}^{K}\sum_{m=1}^{l}\mu\eta P(l,m)x^{l,m}(t)) = \mu\eta P(i,j)x^{i,j}(t)$; and the service received from the second and the third sources are $\frac{x^{i,j}(t)\sum_{l=1}^{K}\sum_{m=1}^{l}\mu(1-P(l,m))x^{l,m}(t)}{\sum_{l=1}^{K}\sum_{m=1}^{l}x^{l,m}(t)}$ and $\frac{x^{i,j}(t)\sum_{l=1}^{K}\mu y^l(t)}{\sum_{l=1}^{K}\sum_{m=1}^{l}x^{l,m}(t)}$ respectively. Based on the analysis, we have the following fluid model:

$$\begin{cases} \frac{dx^{i,1}(t)}{dt} = \lambda_i - \mu\eta P(i,1)x^{i,1}(t) - S^{i,1}(t) \\ \frac{dx^{i,j}(t)}{dt} = \mu\eta P(i,j-1)x^{i,j-1}(t) + S^{i,j-1}(t) \\ -\mu\eta P(i,j)x^{i,j}(t) - S^{i,j}(t) \\ \frac{dy^i(t)}{dt} = \mu\eta P(i,i)x^{i,i}(t) + S^{i,i}(t) - \gamma y^i(t) \end{cases} \quad (5)$$

where $S^{i,j}(t) = \mu\frac{x^{i,j}(t)(\sum_{l=1}^{K}\sum_{m=1}^{l}(1-P(l,m))x^{l,m}(t)+\sum_{l=1}^{K}y^l(t))}{\sum_{l=1}^{K}\sum_{m=1}^{l}x^{l,m}(t)}$, $j \leq i$ and $i,j \in \{1,...,K\}$.

4 Evaluation and Discussion

In this section, we will give a numerical evaluation for various multiple file downloading schemes. From the analysis results, we could find that under the multi-torrent downloading scenario, MTSD outperforms MTCD when the files being requested are interest-correlated. We also find that under the multi-file torrent downloading scenario, by allowing peers to collaborate in their downloading processes, the scheme of CMFSD improves the performance of MFCD greatly, given that the files published within a single torrent are highly interest-correlated.

4.1 File Correlation

In [4], it is reported that more than 85% percent of the peers are found to have joined multiple torrents, and the average number of torrents each peer has joined during its lifetime is 7.514. Although it is widely observed that peers request multiple files, however, from current available measurement results, it is still not clear how many users will request the same set of the files simultaneously under the multi-torrent downloading scenario. On the other hand, under the multi-file torrent downloading scenario, since the files shared within a single torrent are usually highly correlated, most of the peers will choose to download all the files shared within the torrent.

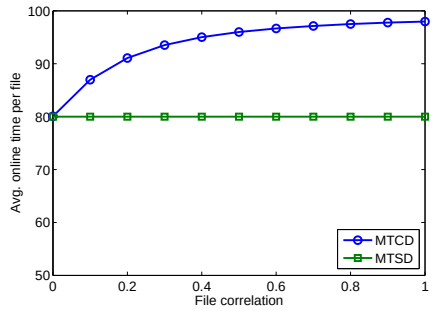


Figure 2. The average online time per file with varying file correlations under MTCD and MTSD. We assume $K = 10$, $\mu = 0.02$, $\eta = 0.5$, and $\gamma = 0.05$ for the fluid model.

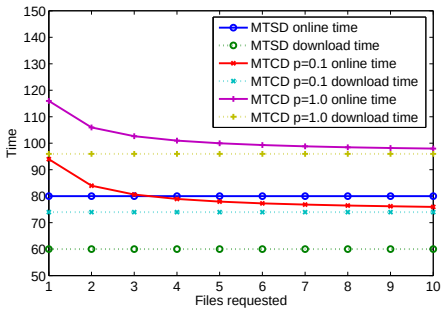


Figure 3. The online time per file and download time per file for peers in different classes in MTCD and MTSD with file correlation of $p = 0.1$ and $p = 1.0$. We assume $K = 10$, $\mu = 0.02$, $\eta = 0.5$, and $\gamma = 0.05$ for the fluid model.

For evaluation of the downloading schemes, we develop a simple model to describe the peers' download behaviors. Suppose for the server-torrent system, there are K files served in the system, these K files could be served by separate torrents as in the multi-torrent downloading scenario, or they could be served in a single torrent as in the multi-file torrent scenario. A user who has requested some files from the system will be likely to request one more file of the K files at probability p. If the web server's user visiting rate is λ_0, then, the users who will request i files enter the system at a rate of $\lambda_0 \binom{K}{i} p^i (1-p)^{K-i}$.

Although this model is very simple, we believe it still could capture some essence of the users' behaviors under the multi-torrent downloading or the multi-file torrent downloading scenario. For example, it is reasonable to assume that p should be usually very small under the first scenario, since the probability that peers downloading the same set of files simultaneously is small (however, we believe that for some types of the content, such as TV plays in separate torrents, this value may also be very large); while under the second scenario, we believe p is near 1 since most of the peers will choose to download all the files in a single torrent. Below we will examine the performance of different download schemes with the file correlation model.

4.2 Evaluation Results

4.2.1 MTCD vs. MTSD

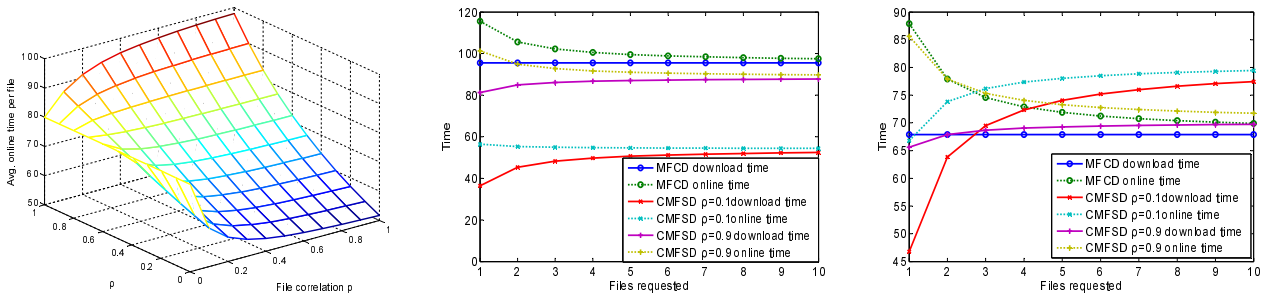
We study the two schemes for multi-torrent downloading schemes, MTCD and MTSD, in this part. We use the *average online time per file* as our main metrics for evaluating the performance of a multiple file downloading scheme. It is calculated as the sum of the online time for all the peers divided by the total number of files the peers have requested.

Based on our model of file correlation, for the MTCD scheme, the entry rate for the ith class peers in a torrent could be calculated as $\lambda_j^i = \lambda_0 \binom{K-1}{i-1} p^i (1-p)^{K-i}$, in the fluid model of (1). Applying the results in (2) and (4), Figure 2 plots the average online time per file under the MTCD and MTSD with various file correlation. We could see that MTCD has a similar performance compared with MTSD when the files have little correlation, but its performance worsens with the increasing of the correlation among the shared files. In Figure 3, the average online time per file and the average download time per file for peers in different classes are plotted under MTCD and MTSD with a file correlation of 0.1 and 1.0 respectively. It is observed that under MTCD, peers requesting more files will have a better performance, and in conditions of low file correlation, MTCD will make a lower average online time for the peers requesting multiple files, but the online time for the majority of peers requesting only one file is longer than that of MTSD; however, when the file correlation is high, both the online time per file and the download time per file for MTCD are longer than those of MTSD as shown in the figure. For average download times, both schemes maintain fairness among the peers in different classes.

Based on the above observations, we conclude that although concurrent downloading is popular in practical applications, the performance of MTCD is worse than MTSD, especially when the files requested are highly interest-correlated. We have two suggestions to avoid this low-efficiency scheme: 1) users should avoid downloading several files concurrently, but should request them one by one; 2) when a user has requested several files in different torrents, the BitTorrent software should download them one by one instead of downloading them concurrently.

4.2.2 MFCD vs. CMFSD

We study the two schemes for multi-file torrent downloading schemes in this part. Note that under the file correlation model, $\lambda_i = \lambda_0 \binom{K}{i} p^i (1-p)^{K-i}$ in the fluid model of CMFSD (in (5)). In Figure 4 (a), the average online time per file under CMFSD is plotted. We vary the file correlation p and the peer's bandwidth allocation ratio ρ from 0.0 to 1.0. It is observed that under any file correlation conditions, setting ρ to 0.0 will have the best system performance, and the performance improvement is more obvious for systems with a high file correlation p. We find that for the extreme case when peers do not allocate any bandwidth for the virtual seeds ($\rho = 1$), the system performs as in MFCD, which has



(a) (b) (c)

Figure 4. (a) The average online time per file with varying file correlations under CMFSD. The online time per file and download time per file for peers in different classes under CMFSD with $\rho = 0.1$ and $\rho = 0.9$, and under MTCD (b) when file correlation of $p = 0.9$ and (c) $p = 0.1$. We assume $K = 10$, $\mu = 0.02$, $\eta = 0.5$, and $\gamma = 0.05$ for the fluid model.

the longest average online time, especially in the high correlated multiple file downloading scenario. In Figure 4 (b) and (c), the average online time per file as well as the average downloading time per file for peers of different classes are plotted under the file correlation of $p = 0.9$ and $p = 0.1$ respectively; we study two polarized cases of $\rho = 0.1$ and $\rho = 0.9$ under CMFSD for each condition, and we also plot the results of MFCD for comparison. Unlike the scheme of MTSD, it is clearly observed that an unfairness in download time per file for peers in different classes exists: peers requesting only one file download faster than peers requesting multiple files, and this unfairness is getting more obvious under the condition that the value of ρ is large ($\rho = 0.9$) and the file correlation is low ($p = 0.1$). Based on these results, we conclude that the scheme of CMFSD could improve the system's as well as the individual user's performance for the multi-file torrent downloading scenario with high correlation among the files, but will introduce unfairness among the peers in different classes, especially when the file correlation is low. However, we could see that under the high file correlation situation ($p = 0.9$), the unfairness is not obvious, and when setting the bandwidth allocation ratio ρ low ($\rho = 0.1$), peers in all the classes get greatly improved performance.

4.3 Deployment of CMFSD

As we have observed in Figure 4, although setting ρ small will improve the system's overall performance, it will introduce unfairness in the download time and the online time between the peers of different classes. When the file correlation is high, this unfairness is acceptable since setting ρ small will benefit all the peers; but in occasional situations when the files in a single torrent have low correlation, this unfairness will make the peers requesting more files having no improvement or even sacrifice on their performance, although the system's average performance is improved.

Another problem arising from the unfairness is that since peers requesting one file will always have a shorter download time compared with peers requesting multiple files, there is an incentive for multi-file requesting peers to pretend to be peers requesting one file. In this case, the cheating peers refuse to upload chunks of the files they have finished via the virtual seeds and set their individual bandwidth allocation ratio ρ as 1. If we view the cheating peer as a special kind of peer: once it has finished downloading a file, it quits and rejoins the torrent with a new ID. This selfish behavior will make the system's file-correlation appeared lower than the real value, and will recursively aggravate the unfairness. If the portion of the cheating peer is large enough, like the low file-correlation situation, the obedient peers' performance will also be seriously sacrificed.

Since an individual peer only cares for its own performance, choosing a reasonable individual value of ρ according to the file correlation, which is a system parameter, as well as other peers' behavior, is necessary. We propose an **Adapt** mechanism for individual obedient peer for this purpose. In the **Adapt** mechanism, a newly joined peer requesting multiple files in the multi-file torrent will set its ρ to 0 initially, which is the best choice for system performance; when the peer begins to serve as a partial seed, it monitors its bandwidth used for uploading the chunks via its virtual seeds and the downloading bandwidth for chunks it has received from other peers' virtual seeds, and calculate the difference between the two as Δ. If Δ is consistently larger than some threshold ϕ_1, the peer increases ρ with a step υ_1, in order to protect the benefit of itself; on the other hand, if Δ is consistently smaller than another threshold $\phi_2 (\phi_1 \leq \phi_2)$, the peer decreases its ρ with another step υ_2 for improving the system's overall performance. The **Adapt** mechanism will be executed periodically until the peer has finished all the files it has requested.

One of the key features of the **Adapt** mechanism is the self-adaptiveness: a peer could probe a better setting of its individual bandwidth allocation ratio ρ by observing its contribution to and benefit from the whole system. When the file correlation is low or the majority of the peers are selfish, with

the **Adapt** mechanism, all the obedient peers will finally set their bandwidth allocation ratio ρ as 1 , and the system works similarly as in MFCD.

Finally, we believe an initial setting of $\rho = 0$ for all the peers joining a multi-file torrent and playing CMFSD is a good choice. There are two reasons: first, we believe that all the files provided in the single torrent are highly correlated, and most of the peers will download all of them; second, peers in real life would be likely to obey the CMFSD protocol, since cheating will cause all the other peers running **Adapt** and the whole system to degenerate into a low efficient mode of MFCD.

5 Conclusion and Future Work

In this paper, we have systematically studied the performance of BitTorrent under two representative multiple file downloading scenarios: the multi-torrent downloading scenario and the multi-file torrent downloading scenario. Particularly, we analyze the schemes of multi-torrent concurrent downloading and multi-torrent sequential downloading, based on the fluid model. We find that when the files are highly interest-correlated, the latter will have a better overall performance. We also consider the multi-file torrent concurrent downloading scheme under the second scenario, and propose a collaborative multi-file torrent sequential downloading scheme. We show via a fluid model based analysis that when the files have high correlation among themselves, the user performance could be improved substantially by our scheme. Finally, a self-adaptive mechanism **Adapt** is proposed for the deployment of the collaborative multi-file torrent sequential downloading scheme by individual peers in a distributed fashion.

For future work, the effectiveness of the **Adapt** mechanism needs to be systematically evaluated, probing the proper settings for the parameters ϕ_1, ϕ_2, υ_1, and υ_2. Another interesting topic is to measure in what scale the files are correlated in the multi-torrent downloading scenario, and how many users would like to download these correlated files concurrently.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," In *Proc. of Workshop on Economics of Peer-to-Peer Systems (P2PEcon'03)*, Berkeley, CA, June 2003.

[2] M. Izal, G. Urvoy-Keller, and et. al., "Dissecting bittorrent: Five months in a torrent's lifetime," In *Proc. of Passive & Active Measurement Workshop (PAM'04)*, France, April 2004.

[3] J. Pouwelse, P. Garbacki, D. Epema and H. Sips, "The bittorrent P2P filesharing system: Measurements and analysis," In *Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Ithaca, New York, Cornell University, February 2005.

[4] L. Guo, S. Chen, and et. al,. "Measurement, analysis, and modeling of BitTorrent-like systems," In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'05)*, Berkeley, CA, October 2005.

[5] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving BitTorrent performance," In *Proc. of IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[6] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," In *Proc. of IEEE Infocom 2004*, Hong Kong, China, March 2004.

[7] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent like peer-to-peer networks," In *Proc. of ACM SIGCOMM 2004*, Portland, OR, August 2004.

[8] L. Massoulié and M. Vojnovic, "Coupon replication systems," In *Proc. of ACM SIGMETRICS 2005*, Banff, Alberta, Canada, June 2005.

[9] Y. Tian, D. Wu, and K.-W. Ng, "Modeling, analysis and improvement for BitTorrent-like file sharing networks," In *Proc. of IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[10] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," In *Proc. of IEEE Infocom 2004*, Hong Kong, China, March 2004.

[11] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," In *Proc. of IEEE Infocom 2005*, Miami, FL, March 2005.

[12] S. Iyer, A. Rowstron and P. Druschel, "SQUIRREL: A decentralized, peer-to-peer web cache," In *Proc. of 21st ACM Symposium on Principles of Distributed Computing (PODC'02)*, Monterey, CA, July 2002.

[13] F. Clévenot and P. Nain, "A simple model for the analysis of the Squirrel peer-to-peer caching system," In *Proc. of IEEE Infocom 2004*, Hong Kong, China, March 2004.

[14] BitTorrent@China. `http://www.btchina.net`.

[15] BitTorrentSpy. `http://www.torrentspy.com`.

[16] K. S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications, Second Edition*. John Wiley and Sons, New York, 2002.