國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management

College of Management

National Taiwan University

Master Thesis

BitTorrent 在多檔案下載環境中的效能分析

On The Performance of Multiple-File Downloads in BitTorrent Systems

俞敦平

Tun-Ping Yu

指導教授：莊裕澤博士

Advisor: Yuh-Jzer Joung, Ph.D.

中華民國 102 年 1 月

January, 2013

致謝

雖然比別人多花了點時間，但終於可以寫謝詞了。經歷了人生第一次的留級，看著同屆的同學們一個又一個的畢業離校，男同學們一個一個的報效國家入伍去了，女同學一個一個進入社會開始工作，還留在實驗室裡的我終於也等到了這麼一天。當口試完成的那一剎那，我這輩子第一次感受到什麼叫做如釋重負，這感覺!真的是太棒了!

回想起來，有些的諷刺，當年千方百計的想要近來，如今絞盡腦汁想要離開。但是這兩年半來確實學到了不少，也經歷了種種，最深刻的當然是記憶猶新的那延畢的半年，感謝各位兄弟們的鼓勵 KE 砲哥、林廢、黃 Sir、啟祥大大、陳好人和賴博。沒有你們的鼓勵與陪伴，儘管嘲諷的多打氣的少，但我一定沒辦法如此坦然的整天窩在實驗室；沒有你們每天幫忙打首勝，我一定沒有辦法心無罣礙的專心完成論文。非常感謝你們。

在這裡最最感謝的當然是教導過我的老師們，尤其是指導教授莊裕澤老師，由於種種原因這兩年半來我在研究與學習上確實非常迷惘，我也自知我在研究上的表現並不理想，非常非常的感謝莊老師的耐心，願意指導並且督促著我這麼一個缺乏幹勁的學生直到我終於完成了論文。也要特別感謝林永松老師和蔡益坤老師在口試時對我的研究所進行的提醒和指導。非常感謝你們。

感謝我的父母們，非常非常的感謝，沒想到你們願意容忍一個研究所讀了兩年半的阿宅兒子，感謝你們持續的支持和鼓勵並且沒有把我掃地出門。我不可能完成這個學位。非常感謝你們。

非常非常感謝我的表弟何睿(Ray Ho)。他在必須同時兼顧學業和工作的情況下，不只幫忙修改我論文中的文法錯誤，更大幅修改幫忙將我論文中不怎麼流暢的中式英文句子，修改成較為通暢且正規的英文寫法。更讓我感動的是儘管我們 25 年來只見過三次面，他仍然在短短的兩天內就將我整篇論文大幅度的修改完成，對於最後趕著畢業入伍的我實在是幫助的太大太大。非常感謝你。

要感謝的人實在太多，想說的話也非常的多，礙於篇幅只能最後只能簡單感謝所有我在台大這六年半來愈到的所有人物，真的非常感謝大家。

研究生　俞敦平

January, 23 2013

論文摘要

作者:俞敦平　　　　　　　　　　　　　　　　　　　　民國一零二年一月
指導教授：莊裕澤博士

BitTorrent 在多檔案下載環境中的效能分析

BitTorrent(BT)是一種點對點(peer-to-peer)的檔案傳輸協定，由於其獨特的設計在傳輸效能上有著非常優異的表現，因此常常被用於在網路上散佈大型檔案或用於高畫質的影音串流傳輸，BT 也是目前世界上流量最大的網路協定之一，也因此吸引了不少學者針對 BT 這個協定進行了大量的研究。

其中 BT 最為驚人的一項特性也就是其下載時間不受使用者進入系統的速率的特性，造就了其獨特的可擴展性(scalability)，更讓 BT 一時之間成為了被大家爭先研究的對象。但這些研究多數是爭對單一檔案下載(single-file download)情況下的研究，儘管有少部分關於多檔案傳輸環境的研究，但仍然缺乏對於多檔案下載(multiple-file download)環境統一的整理和分析。並且根據觀察實際世界中的 BT 使用情形，超過 85%的使用者其實同時間都在傳輸複數的檔案也就是都處於多檔案傳輸的情況下。

我們的研究發現在網路頻寬的效用(utilization)很高的情況下，所使用不同的多檔案傳輸方式，例如 MFMT(Multiple-File-Multiple-Torrent)、MFST (Multiple-File-Single-Torrent)或是針對 MFST 的情況下增加所使用的上傳連結(unchoking slot)數量等等多種方法再無論檔案大小是否均一的情況下效能都是相同的。另外我們也在實驗中發現都整體網路的頻寬較小時，使用者進入系統的速率越高的時候，系統平均的下載時間會變得更長。

關鍵字：BitTorrent、多檔案下載、上傳連結數量、下載效能、檔案大小差異

THESIS ABSTRACT

Student：Tun-Ping Yu January, 2013
Advisor：Yuh-Jzer Joung, Ph.D.

On the Performance of Multiple-File Downloads in BitTorrent Systems

BitTorrent is a peer-to-peer file sharing protocol; due to its unique mechanism, it has excellent file transport performance, so it is widely used to distribute large amounts of data across the Internet and to stream high-quality video. BitTorrent is also one of the largest traffic consumers on the Internet, attracting various research on this protocol.

One of the most amazing characteristics of BitTorrent is that the average time a peer stays in the system is not related to the entry rate of peers into the system. This indicates that BitTorrent has great scalability. This characteristic attracts even more research on this protocol, but most of it focuses on the theme of single-file downloads. Research on multiple-files downloads exists, but there is still a lack of rigorous, organized research on the topic. According to real-world traces, over 85 percent of BitTorrent users download multiple files concurrently.

Our research found that when the bandwidth utilization is high, different multiple-file-download approaches—such as MFMT(Multiple-File-Multiple-Torrents)， MFST (Multiple-Files-Single-Torrents) or MFST with extended unchoking slots have similar average download times, regardless of whether of file sizes are unique or not. We also found that the entry rates of peers will actually affect the average download time of peers when the system is operating in a low bandwidth environment.

keywords：BitTorrent、Multiple-file-download、unchoking slots number、performance 、file size difference

Contents

List of Figures

List of Tables

Chapter 1

Introduction

BitTorrent [1], also known as BT, was first introduced by Bram Cohen in 2003 [2]. Since then, it has become one of the most popular and widely used P2P file sharing protocols on the Internet. According to Sandvine (Fall 2011) [3], BitTorrent still accounts for 16.5% of Internet traffic in North America, second only to Netflix. It is also the only major P2P network currently ranking as one of the top ten Internet traffic consumers in North America.

In a P2P file sharing network, peers in the system not only act as a client but also as a server. Instead of downloading from a centralized server, files are downloaded from other peers in the P2P network. This way, every peer contributes to the system and the workload is distributed across the whole network. The mechanism mentioned above is the reason why P2P file sharing systems are very scalable.

In the following section, we will provide background by briefly describing BitTorrent protocol and its mechanism.

1.1 Background

To download files with BitTorrent, a peer needs to download a specific metafile called a *Torrent*. Torrents are usually downloaded from web-based forums, which are called *Portals*. Compared to other P2P file sharing protocols that have provided searching function to locate the file such as *eDonkey* and *Foxy*, BitTorrent seems inconvenient without built-in searching functions. However, it is still one of the most popular—if not the most popular—P2P file sharing protocol because its well-designed mechanisms make it extremely efficient, especially when sending large amounts of data across the network.

BitTorrent divides files into smaller pieces. As soon as a peer finishes downloading a complete piece, it is able to contribute to the system by uploading that same piece. The combination of the above-mentioned techniques with the *piece selection* and *choking* algorithms forms the core of BitTorrent's mechanisms. They are the reason for the extreme success of BitTorrent. We will further discuss these mechanisms in chapter 2. In the next section, we are going to state our motivation.

1.2 Motivation

Due to the extreme success of BitTorrent, there has been plenty of research on it. Those works have revealed the characteristics and performance of BitTorrent [4-6].

Most of those studies focus on a single-torrent system. They assume a node joins a single torrent, one at a time. However, another study shows that more than 85 percent of peers concurrently join multiple torrents [7]. Even though there is little research about downloading multiple files in the BitTorrent system, some of research has proposed several techniques for collaboration among multiple torrents to improve the performance of the BitTorrent system under multiple-file downloading schemes. To the best of our knowledge, due to the difficulties faced in deployment and implementation, current practical BitTorrent clients do not utilize the technique of working with collaboration between different torrents.

Practically, not only do peers tend to join more than one torrent at a time, a single torrent can contain multiple files [8, 9]. For example, different episodes of a TV show or different songs by the same singer are often published together. Peers tend to download those highly correlated files together. Here we find an interesting question that does not yet have an answer. When publishing multiple files, assuming the files are all highly correlated so that all peers will download all of them: I Is it more efficient to put all of those files into a single torrent, or is it more efficient to place them into different torrents?

Another question arises from the scheme above. When multiple files are published within a single torrent, most practical BitTorrent clients only use a single session to handle it, and unchoking slots are set within a session. For example, if three files are packed into a single torrent, the BitTorrent client will not use three times the unchoking slots downloading them. We are wondering if this could be a performance factor in a multiple-file download environment.

Most of the practical BitTorrent clients use 5 *unchoking* slots per torrent. This means for each torrent a peer will simultaneously upload to at most 5 other peers. There is some research about the unchoking mechanisms of BitTorrent. Fan et al. [10] give a thorough insight into the trade-off between two different strategies applied to choking slots. Laoutaris [11] proposed an uplink allocation algorithm that they called "BitMax". The BitMax algorithm focuses on improving overall performance by increasing uplink utilization. Uplink utilization is achieved by dynamically changing the unchoking slots number and the uplink allocates to each unchoking node. But there still an unanswered question: When downloading multiple files within a single session, does opening more unchoking slots according to the number of files improve overall performance?

These problems, commonly faced by a BitTorrent user, need almost little modifications to apply those changes. Our goals are to discover the influence of packing files together under different downloading environments and the effect of changing unchoking slots under different conditions. We believe our research could really help BitTorrent users to adjust their own client settings.

1.3 Research Objectives

We first try to analyze the different schemes mentioned above with a fluid-based model [4] and try to understand their performance under the steady-state. Even though fluid-based mathematical models can capture fairly well the characteristics of the BitTorrent system, the model is limited to describing the system performance under the steady-state.

According to trace analysis of [9, 12] , in the real-world BitTorrent system the stable period is actually very short, so we also want to know about the overall performance of the system over its entire lifetime. As a result, we decided to use the simulation-based study as another way to observe the performance and evaluation of the BitTorrent system under non-steady-state conditions.

The simulator we used is *General Peer-to-peer Simulator* (GPS) [13]. GPS models the full behavior of the BitTorrent protocol. We believe it could very well help us evaluate the conditions of the BitTorrent system.

Chapter 2

Related Works

Bram Cohen revealed BitTorrent in 2003 [2]. Because of its wide deployment and extreme success, there have been various studies performed on the measurements, modeling, and algorithms of BitTorrent systems. In this chapter we first present an overview of BitTorrent mechanisms and establish the terminologies in section 2.1. In section 2.2 we discuss various works on the BitTorrent System to provide an understanding of BitTorrent performance and its characteristics. In section 2.3 we focus on research related to multiple-file download systems. Finally, in section 2.4 we will have a short summary at the end of this chapter.

2.1 BitTorrent overview

We will start by introducing the terminologies and basic concept of BitTorrent in 2.1.1 then further discuss the mechanisms used in BitTorrent in 2.1.2.

2.1.1 Terminologies

In the BitTorrent system, files are first divided into smaller file chunks called *pieces*. Every piece is divided equally except for the final piece. A peer who wants to upload a file must first create a small metafile called *a torrent*. The torrent file holds the URLs of *trackers*, file size, file name and cryptographic hash(SHA1) of all pieces. The length of the piece is usually a power of 2, and size is chosen based on the file size. Smaller piece size usually has better efficiency but also will increase the size of the *.torrent* meta file. The most common and practically used piece sizes 256KB, 512KB, and 1MB [14]. Usually torrent files are published and stored on a specific website called *a portal*, such as “The Pirate Bay [15]” to allow other users to download them.

Peers join the same torrent and form a swarm. For a node, every torrent they joined is called a session. Those who have finished downloading and are still willing to stay in the swarm uploading the files are called seeds. Those who have not yet finished downloading are called leechers. The tracker is a centralized component that assists the communication between peers. The tracker maintains the list of peers participating in the same swarm. Every peer initiates the download by requesting a peer list from the tracker. Peers that are currently downloading can also periodically update the peer list from the tracker. The tracker can also gather statistical information from peers who are willing to provide their stats; these tracker logs and traces are valuable research resources. Trackers only help peers discover each other, but do not assist in any form of file transfer.

A technique based on *distributed hash table* is implemented to provide support for “trackerless” download, which allows peer downloads from a torrent without a working tracker. Another way to gather information from peers in the same swarm is through *Peer exchange (PEX)* Message. Peer exchange allows peers to exchange the peer list with their connected peers.

2.1.2 BitTorrent mechanisms

The following introduction about BitTorrent are reference from Bram [2] and BitTorrent protocol specification [14].

● Piece selection

The strict policy of piece selection is that if any sub-piece has been requested, the remaining sub-piece will be requested first. This makes sure the complete piece can be finished as soon as possible.

At the start of download, a newly incoming peer has no piece to upload. It just tries to get a complete piece as soon as possible. So the newly joining peer just randomly selects a piece to download until it has a complete piece.

The most important piece selection mechanism and the one which makes BitTorrent extremely efficient and available is the *Local Rarest First* (LRF). Peers are have the knowledge of which pieces their connected peers get. Local rarest first means peers tend to download the fewest pieces distributed among the connected peers. This technique does a great job of evenly distributing the pieces throughout the swarm, and also prevents the existence of a particular piece that no longer exists in the system after the original seed leaves.

There is also *endgame mode*. When all sub-pieces that have not finished are being actively requested, a peer will send requests for all sub-pieces to all connected peers. The purpose of endgame mode is to avoid the potential delay in completing a download caused by requesting the last few pieces from peers with slow upload bandwidth.

- Choking algorithm

In BitTorrent, when you upload a file to a specific peer, you are *unchoking* that peer. And *choking* a peer means you do not upload to that peer.

BitTorrent is without central resource control. Every peer just tries to maximize their own download rate. The basic concept of a choking algorithm is a *Tit-for-tat* (TFT) strategy. Tit-for-tat means that a peer simply uploads to those who upload the most to them. Usually a peer is simultaneously uploading to 5 other peers. Four of the unchoking slots apply the TFT strategy. For every 10 seconds, a peer will select 4 peers who have uploaded the most to it during the past 20 seconds [11].Then, the fifth slot applies a different strategy called optimistic unchoking. For every 30 seconds the peer randomly unchokes a peer it is connected to. This lets the peer itself search for better peers to connect to. This also helps new incoming peers to get their first piece. Qiu and Srikant [4] prove TFT is very efficiency through a mathematical model.

When over a minute passes without getting a single piece from a particular peer, the peer is *snubbed* by that particular peer. An *anti-snubbing* strategy in BitTorrent will prevent the peer from unchoking those who snubbed you except through the optimistic unchoking. Also when a peer is *snubbed* from all peers it currently downloads from, it will temporarily generate multiple optimistic unchoking slots for exploring better peers to download from.

- *Queuing* and *Super Seeding*

To prevent performance loss due to high round trip time between receiving a piece and the next request message, a peer usually keeps a few unfulfilled requests on each connection.

Super seeding is not part of the original BitTorrent protocol and it is only for the original file distributor. When the original file distributor chooses to use super seeding mode, it will pretend itself as another leecher in the system. When peers connect, the seed will inform a peer that received a new piece and that particular piece will be the one that has not yet been sent to anyone. The seed will not upload another piece to the same piece until the seed found another peer that has a particular piece. Super seeding help peers to download only the rarest data and reduces the amount of upload performed by original seeds before other clients turn into seeds. Upon finishing the uploading of one complete copy of the file using super seeding mode, multiple new seeds will emerge in a short period of time, thus creating a big boost for the overall performance of the swarm.

2.2 BitTorrent system

Due to the success of BitTorrent in practice, plenty of research tries to capture, analyze and study its success through theory. Mathematic modeling becomes a very popular way to capture the characteristics of the BitTorrent system. We will first discuss them in 2.2.1. However, due to the extreme complexities of the BitTorrent system, some abstraction of the details or unrealistic assumption are made on those models, limiting those models to mostly describing the system under certain situations. As a result, other research is trying to capture a more realistic view of the BitTorrent system by using simulation-based approaches or analyzing the logs and tracker traces. These are going to be discussed in 2.2.2.

2.2.1 Analysis based on Mathematic modeling

Qiu and Srikant [4] present a fluid model for the evolution of peers number, with the assumption that peers arrive following a Poisson process. By using the fluid model above and applying the Little's law, the average download time for a peer under steady-state was calculated. The result shows that under steady-state the average download time of a peer is not related to the arrival rate of incoming peers, which means that the BitTorrent system has tremendous scalability. Authors also present a simple model showing that file sharing is very effective in BitTorrent.

Fan et al. [10] characterize the design space of BitTorrent protocols. They consider BitTorrent's two design objectives: 1) Performance—minimizing the average download time; 2) Fairness—those who contribute more should receive better service than those who contribute less.

They consider the scenario in which file size is uniform but peers have heterogeneous bandwidth. They neglect the bottleneck caused by network topology. They assume that only the upload and download bandwidth of peers create the limit to the download speed. It is because current the BitTorrent system has no incentives for seeding. They also assume that peers leave the system right after they finish the download. Using the mathematical model they present, the fundamental trade-off between performance and fairness of BitTorrent protocol is explored. The influence of numbers of regular choking slots and optimistic unchoking slots on the average download rate and fairness are modeled. They found that higher ratios of optimistic unchoking slots improves the overall performance of the system and higher ratios of regular unchoking slots improves the fairness of the system. Authors also mentioned that in most practical BitTorrent clients four regular unchoking slots and one optimistic unchoking slot are used, which indicates that most practical BitTorrent clients focus more on fairness than performance.

There are also works like[11] [16, 17] that present additional models for more complicated situations and different issues like fairness and availability.

2.2.2 Measurement and simulation based studies

Izal et al. [12] studied the trace of a tracker's log during a 5-month period. The authors showed that BitTorrent performs pretty well and is very effective in distributing large content simultaneously to the extensive amount of nodes. They also point out that BitTorrent is realistic and inexpensive compared to traditional ways of distributing files through FTP or HTTP. By combining the piece selection mechanism and the choking algorithm BT is extremely effective and efficient [18].

Pouwelse et al. [19] present a measurement study of BitTorrent based on trace gathered over a period of 8 months from a single torrent –search site. Issues like availability, integrity, flash crowd handling, and download performance are studied. They also point out that currently, BitTorrent lack incentives to seed.

Zhang et al. [20] also provide a nearly complete picture of the entire BitTorrent ecosystem through measurement studies of traces crawling from five of the most popular torrent-discovery sites and Azureus and Mainline DHTs. Issues like torrent-discovery, tracker, peer, user behavior and content landscapes are studied.

Bharambe et al. [6] study BitTorrent with flash crowds by using simulation. They discovered that BitTorrent can achieve high uplink utilization but could be unfair to peers with high bandwidths.

Other measurement studies [9, 21] combine with the above-mentioned research to bring great insight on the BitTorrent system.

Most of these studies focus on single-torrent system, but in practice most peers join multiple torrents and download multiple files at a time. We are going to discuss those works on downloading multiple files in section 2.3.

2.3 Multiple files download

2.3.1 Analysis and modeling based on real-world trace

Guo et al. [9] analyze and model BitTorrent traffic based on trace from the real world. According to the trace they found that over 85% of users participate in multiple torrents.

They present a model to describe the evolution of a single torrent during its entire *lifespan*. They model and study the client performance variations and service fairness in the single-torrent system. They found that the client-performance fluctuates in a single-torrent but is quite stable when aggregated over multiple torrents. They also observed that the BitTorrent seeds service policy has a lack of incentives for seeds to contribute.

They are motivated by observation of the single-torrent system. They also present a study of multiple torrents through modeling and trace analysis. Their models assume that peers are willing to seed and delineate a peer's life cycle into a sequence of *downloading*, *seeding* and *sleeping*. They also assume that each peer joins each torrent at most once, and joins one torrent at a time.

From their studies, they proposed an inter-torrent collaboration mechanism that enabled exchange-based incentives among multiple torrents. According to their model, the inter-torrent collaboration mechanism can significantly reduce the failure ratio of a single torrent, but their method needs the assistance and modification of the current tracker. That is definitely quite challenging in real-world deployment.

2.3.2 Modeling of multiple-file downloads

Tian et al. [8] present a fluid-model based analysis on different multi-torrent schemes. In a Multi-torrent concurrent downloading (MTCD) scheme, the user enters into separate torrents for multiple files. It is applied in most practical BT clients. In the MTCD scheme each session is competing with each other for the available bandwidth.

In multi-torrent sequential downloading (MTSD), the user only enters one session at a time. Multi-file torrent concurrent downloading (MFCD) means a single torrent contains multiple files, and for each separate file, the user uses an independent session to download them concurrently. In their model MTCD and MFCD are treated the same. But they still mention that in most cases files published within a single torrent are usually highly interest-correlated.

Using *average online time per peer* as the main evaluation metrics, MTSD outperforms MTCD when correlations between files are high. But MTCD has a similar performance with MTSD when files have little correlation. Based on the above observation from their model, they also proposed a *collaborative multi-file sequentially downloading* scheme (CMFSD).

In CMFSD a peer downloads multiple files sequentially. They treated every individual file in a single torrent as a small *sub-torrent.* A peer will have two sessions instead of one at a time. One session sequentially joins those sub-torrents as a leecher. Another will join a finished *sub-torrent* seeding. They also provide an adapt mechanism for deployment of CMFSD.

Through the fluid model, although they demonstrate when files are highly correlated, their scheme could greatly improve the performance of the system. They claimed their adapt mechanisms would prevent peers from trying to cheat, but it still does not guarantee that a peer is willing to seed in a finished file. Also their approach requires the modification of current BitTorrent protocol.

2.3.3 Simulation-based studies of *Cross torrent tit-for-tat*

Yang et al. [22] proposed a *"cross-torrent-base" tit-for-tat* (CTFT) strategy as a way to provide incentives for nodes to act as seed. CTFT modifies the unchoking algorithm of leechers. Instead of choosing the peers who upload the most in a particular torrent, they consider the aggregate upload in all torrents that they both participate in. Incentives for seeding are provided by giving higher weight for uploading received from seeds.

They also proposed another modification of CTFT called *CTFT with dynamic* *seeding* (CTFT-DS). In CTFT-DS each node locally estimates the ratio of seeds to leechers. Then, they only seed in those where the ratio is below a certain threshold. The researchers also present a series of simulation studies.

In their simulation, they assume peers arrive following a Poisson process. Each torrent has an original seed, which will stay for the entire lifetime of the torrent. They consider both a homogeneous system and heterogeneous system. In the heterogeneous system, two classes of peers are defined: a fast peer with higher upload and download rates, and a slow peer with slower download and upload rates. They assume there are more slow peers than fast peers. They also use files from the real world in their experiment. Game patches, software installers and online movie rental systems are used. They also test two conditions: when all peers join the same set of torrents, and when all peers just randomly join 3 torrents out of 10. They use the default setting adapted by most practical BitTorrent client: five unchoking slots with seeds and four plus one optimistic unchoking slots with leechers.

The main metrics they consider are the average download time over all torrents and the average total time a node takes to complete all download. The results show that performance gains are possible through cooperation across multiple torrents. However, their approaches require modification of the current TFT strategy. Also, a malicious peer could take advantage of the system by disguising itself as a seed in some torrents.

2.4 Summary

In section 2.1, we first introduce the mechanisms used in BitTorrent and established the terminology. Then we discussed the various related research of the BitTorrent system in section 2.2. Those works have brought tremendous insight into the overall system performance and fairness. In section 2.3 we focus on the research on multiple-file download environments. They have all proposed several techniques to relate multiple torrents together to improve the overall system performance. However, all of them consider it from the viewpoint of a leecher or from a global view of the entire system. In reality, it is impossible for peers to obtain global knowledge of the entire network. It is also a bit challenging to let every peer in the system apply some modified strategies. Also for the techniques proposed, all of them required some modification to current BitTorrent protocol or BitTorrent client implementation.

Considering things from the leechers' point of view is very reasonable, because most of the peers in the systems are leechers. In [12] and [9] we learn that seeds contribute more than leechers, but surprisingly to the best of our knowledge, none of the research starts from the viewpoint of the original file distributor.

In our work we are trying to find a way to improve system performance without any modification of current protocol or client implementation. Also, from the research above we learned that seeds contribute a lot to the system and most of the research focuses on providing incentives for peers to seed. We start from a different point of view. We consider the impact of the0 way in which the original file distributor distributes the files and its relation to various parameters that can be managed and modified by clients themselves.

Chapter 3

Methodology

There are several ways to download multiple files in BitTorrent. Much research has already tackled this topic. Each researcher has their own assumptions and abstractions of network details and their own solutions to the multiple-file download scenario of BitTorrent. Even in practice, different BitTorrent clients use different approaches to deal with the multiple-file download situations they faced. In this chapter, we are going to summarize and classify the different multiple-file download scenarios in BitTorrent and compare the downloading efficiency between different multiple-file download approaches.

In this chapter we will first discuss some of the past research on multiple-file download in BitTorrent. We then classify the different approaches to multiple-file downloads according to our own knowledge gleaned from previous research. Afterwards, we will compare the download efficiency between different download methods. Finally, we try to discuss some other parameters that might affect the download efficiency under different download approaches, such as number of unchoking slots and the size differences between download files.

3.1 Multiple-File Download in BitTorrent

Multiple-File Download in BitTorrent

When using BitTorrent to share multiple files, the original file distributor can decide whether to distribute those files in multiple torrents separately or pack those files together into one single torrent. We called the former *Multiple-File-Multiple-Torrent* (MFMT) download and the latter one is called *Multiple-File-Single-Torrent* (MFST) download.

One thing that needs to be mentioned here is that in the real world, BitTorrent clients are more likely running multiple tasks and that those tasks can consist of both single-file torrents and multiple-file torrents. But for simplicity, in this research we assume a user will either chose MFMT or MFST as their downloading method when they trying to download multiple files using BitTorrent.

Another thing we need to mention is that even though the original torrent creator decides the number of files that a torrent contains, whether to download all of them can be decided by the downloader. Most practical BitTorrent clients allow the user to pick the files to download even if they are packed into a single torrent. In the real world, files in a single torrent usually are highly correlated [8]. For example the different episodes of same television show or different songs by the same singer are commonly packed into a single torrent, and usually the downloaders tend to download those files together. Compared to a situation where users are downloading multiple files from a single torrent, the chance of users downloading the same set of multiple files from different torrents is very low.

Some past research

Gue et al. [9] find out that at over 85% of BitTorrent users join multiple torrents through trace analysis. Tian et al. [8] compare the performance of two different downloading strategies of MFMT under steady-state. The first strategy is called *Multi-Torrent Concurrent Downloading* (MTCD). MTCD is when a user tries to download multiple files, the user simultaneously downloads all of those files concurrently and the bandwidth is equally shared among all those torrents. The second strategy is called *Multi-Torrent Sequential Downloading* (MTSD). With MTSD, the user sequentially downloads those files one at a time. All of the bandwidth will be used on one single torrent, then switches to the next torrent when the current one is finished.

In Tian's et al. [8] model, they adopt the concept of *File Correlation*. *File Correlation* refers to the probability that a user downloads the same set of files. The definition of file correlation is as follow: Assuming there are *K* files currently in the system, a user will have a probability of *p* to download another file. If the entry rate of all new peers into the system is λ_0, than a peer who is already downloading *i* files will enter the system with the rate of $\lambda_0 \binom{K}{i} P^i (1-p)^{K-i}$.

Tian et al. [8] models that, when *File Correlation* is low, MTCD and MTSD experience similar performance. But with the increase of *File Correlation*, the performance of MTCD will decrease. This is because in his MTCD model, he assumes that a peer who downloads *i* files will only act as seed after every *i* files are finished. In the MTSD model, the users will seed for a period of time after every single file is finished. Also, in his model he assumes that utilization of upload capacity seeds are much higher than leechers, so the difference of performance comes from the different average seeding time in these two methods.

He also mentions the download scenario of MFST, which he calls *Multi-File Torrent Concurrent Downloading* (MFCD). They assume that when a BitTorrent client finds out that there are *n* files in a single torrent, they will divide it into *n* subtorrents, and for each of the subtorrents they will generate a virtual peer to download the file. The bandwidth will equally distribute among all virtual peers just as if they are *n* different torrents. From this point of view his MFCD and MTCD scenario is the same. But he also mentions the cases where usually in MFCD the *File Correlation* is higher than those in MTCD [8, 23].

Although they assume in MFCD that different files in a single torrent actually belong to different swarms, there might be another possible scenario in which BitTorrent clients do not actually distinguish different files within a single torrent. This means that no matter how many files are within the torrent, they will all be downloaded under the same swarm, which is commonly called *Content Bundling*. The *Content Bundling* scenario is actually the same as Single Torrent Single File scenario. They are even the same when BitTorrent clients allowed users to download a selected set of files in the torrent. We just simply need to treat those peers who download only a portion of the torrent as aborted peers, then we can simply just apply the model of Qiu et al. [4] to describe the performance of *Content Bundling* under steady-state.

Menasche et al. [24] analyze the difference between *Content Bundling* and Single File downloading systems from another point of view. They assume the downloads are a process that switches between waiting time (Busy Period) and service time (Idle Period). Waiting time is the time when a peer is waiting for a download; thus service time is the actual time spent downloading the files.

Their models assume BitTorrent download is a coverage process. The first busy period starts when the original publisher enters the system. The lifetime of a swarm is switching between the busy period and idle period. In the model, they prove that bundling several files together can improve the availability of unpopular files in it. They also point out that the download time of an unpopular file is mainly restricted by its waiting time, so bundling those files together or with other popular files can improve their download time efficiently.

Different Multiple-File Download Approaches of BitTorrent

Here we categorized the different multiple-file download approaches of BitTorrent we have mentioned in above chapters.

From downloading methods

There are two download methods to download multiple files in BitTorrent. *Concurrent download* is to simultaneously download all the files at once and *Sequential* *download* is to sequentially download these files one by one in a certain order. In this work, we think of sequential download as a sequence of single-file download and will not discuss it any further. We will mainly focus on *Concurrent download* in this work. *Concurrent download* is actually the default downloading method when downloading multiple files in most of the practical BitTorrent clients. Also, in most commonly used BitTorrent clients, the user is allowed to set the number of tasks that are allowed to run simultaneously. Sequential download is a special case in which only one task is allowed to run at a time. In most practical situations BitTorrent also has built-in scheduling functions that allow users to perform sequential downloads.

Inter-torrent collaboration

Another way of classifying different multiple-download approaches in BitTorrent is to classify them according to the existence of inter-torrent collaboration. We assume that when inter-torrent collaboration is considered the TFT between the collaborated torrents are considered together.

According to Torrents

We can also distinguish the different download approaches by the number of files within the torrent. We only consider two cases, the single-file torrent and the multiple-file torrent.

Even though in the real world, the multiple file downloads usually consist of both *Single-file-torrent* and *Multiple-file-torrent*, for simplicity in this work we only consider cases that only have multiple *Single-file-torrent* and single *Multiple-file-torrent*. We classify the BitTorrent multiple downloads as the following scenarios:

● *Multiple-File-Multiple-Torrent Downloading with Inter-Torrent Collaboration*:

This scenario is when user concurrently downloads multiple *Single-file-torrents*. The inter-torrent collaboration in this work simply assumed that the TFT strategy is considered among all torrents downloads.

● *Multiple-File-Multiple-Torrent Downloading without Inter-Torrent Collaboration*:

This one is same as the previous case, except the TFT are considered separately in each individual torrent.

● *Multiple-Files-Single-Torrent Concurrent Downloading*:

In this case the multiple files are all bundled in a single torrent.

3.2 Multiple-File-Multiple-Torrent

In this section we are going to discuss and analyze the different parameters that may affect download performance under the two different MFMT cases.

3.2.1 MFMT with Inter-Torrent Collaboration

The first case we consider is the case of MFMT with Inter-Torrent Collaboration. Because Inter-torrent collaboration is not actually defined in current BitTorrent specification and not yet used in practical BitTorrent clients, we made the following assumption of the Inter-Torrent collaboration.

- We assume that for all torrents downloaded by the same peer, the TFT are considered together.

With the above assumption, the case of MFMT with inter-torrent collaboration can be simply viewed as a single-file system, because all the files are downloading within the same swarm in the system. In other words, those multiple files can be treated as a single large file, and every individual file can be seen as part of the large file. For the single-file download performance of BitTorrent under steady-state Qiu et al. [4, 6] already has a model that describes it fairly well:

$$\frac{dx}{dt} = \lambda - \theta x(t) - min\{cx(t), \mu(\eta x(t) + y(t))\},$$

$$\frac{dy}{dt} = min\{cx(t), \mu(\eta x(t) + y(t))\} - \gamma y(t), \qquad (1)$$

λ	***The entry rate of peers into the system.***
$x\ (t)$	***The number of leechers in the system time at time t.***
$y(t)$	***The number of seeds in the system time at time t.***
μ	***The upload bandwidth of all given peers.***
c	***The download bandwidth of all given peers. We assume c ≥μ.***
θ	***The rate of the aborted leechers leaves the system.***
γ	***The rate of the seeds leaves the system.***
η	***The effectiveness of file sharing. Values are in [0,1].***

Table 1 Notation used in the MFMT with Inter-Torrent Collaboration model

The average time a peer in the system under steady-state T is,

$$\mathrm{T} = \frac{1}{\theta + \beta}. \quad (2)$$

and $\frac{1}{\beta} = max\left\{\frac{1}{c}, \frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right)\right\}$。The model above revealed some interesting characteristics of the BitTorrent system when downloading a single file.

- The higher the η, the shorter the T. It is pretty obvious that the more effective the file sharing, the shorter the overall download time.

- The longer the average seeding time, the faster the download time.

- The most important characteristic he discovered is that the download time is not relevant to the entry rate of λ. That is the reason why BitTorrent is scalable.

3.2.2 MFMT without Inter-Torrent Collaboration

Here, we try to model the case of MFMT without Inter-Torrent Collaboration. We first make some assumptions to prevent our model from becoming too complicated. For all assumptions below we are considering the situation under the system steady-state.

- We assume all peers are have the same upload and download bandwidth.
- We assume the upload bandwidth is significantly higher than download bandwidth, meaning that upload bandwidth is the only bottleneck to consider.
- We assume there are sufficient number of peers, thus the utilization of upload bandwidth is full.
- There are no aborted peers in the system.
- The bandwidth of a peer is equally shared between all downloading torrents.
- If there are n files in the system, and system is under steady-state, the n files will finished in the order of F_1, F_2,…,F_{n-1},F_n.
- We assume the peers will form n sub system. We define Sub system S_1 as the peers who just enter the system and have not yet finished downloading file F_1. Thus, sub

systems S_i is formed by the peers that have finished downloading all the files from F_1 to F_{i-1}, but have not finished downloading file F_i yet. For example, S_2 consists of the peers that finished downloading the file F_1 but have not yet finished downloading file F_2.

- We assume that when the system is under steady-state, all the sub-systems are also under steady-state.

With the above assumptions, we now list all the notations we used in the following tables.

λ	***The entry rate of peers into the system.***
μ	***The upload bandwidth of all given peers.***
η	***The effectiveness of file sharing. Values are in [0,1].***
N_r	***Number of regular unchoking slots.***
N_o	***Number of optimistic unchoking slots.***
$S_1(t)$	***Number of leechers in the sub system 1 at time t.***
R	***$R = \frac{N_r}{N_r+N_o}$ The ratio of regular unchoking slots.***
F_i	***Size of File i.***

Table 2 Notation used in the MFMT without Inter-Torrent Collaboration model

The evolution of number of peers in sub-system S_i is as follows:

$$\frac{ds_i(t)}{dt} = \lambda - \frac{\eta\mu R\frac{s_i(t)}{2} + (1-R)\mu\sum_{k=1}^{i}\frac{1}{k}s_k(t)\frac{s_i(t)}{\sum_{l=1}^{i}s_l(t)}}{F_i - [the\ part\ of\ F_i\ download\ during\ S_1\ to\ S_{i-1}]}, \qquad (3)$$

$\eta\mu R\frac{s_1(t)}{2}$ is the upload received from the *Regular unchoking* from the peers in sub system S$_i$. $(1-R)\mu\sum_{k=1}^{i}\frac{1}{k}s_k(t)$ comes from the *Optimistic Unchoking* of the other *i* sub-system that are still downloading file F$_i$. $\frac{s_i(t)}{\sum_{l=1}^{i}s_l(t)}$ is the ratio of peers in S$_i$ to all peers that are currently downloading file F$_i$. Because of the unbiased feature of *Optimistic Unchoking*, the total bandwidth received from *Optimistic Unchoking* of peers in S$_i$ is calculated as the total bandwidth from all files that are currently downloading F$_i$ times the proportion of peers in S$_i$ in all peers downloading F$_i$ $(1-R)\mu\sum_{k=1}^{i}\frac{1}{k}s_k(t)\frac{s_i(t)}{\sum_{l=1}^{i}s_l(t)}$.

To study the performance under the steady-state, we let

$$\frac{ds_i(t)}{dt} = 0.$$

From 3 we get

$$0 = \lambda - \frac{\eta\mu R\frac{\overline{s_i}}{2} + (1-R)\mu\sum_{k=1}^{i}\frac{1}{k}s_k(t)\frac{s_i(t)}{\sum_{l=1}^{i}s_l(t)}}{F_i - [the\ part\ of\ F_i\ download\ during\ S_1\ to\ S_{i-1}]}, \qquad (4)$$

But here we are unable to find a general solution for the value of

$[\textit{the part of } F_i \textit{ download during } S_1 \textit{ to } S_{i-1}]$.This need to calculate step by step from S_1, and the models get more complicated during the process. We will do an example of how to calculated the average download time under the special case of n=2.

3.2.3 MTCD without Inter-Torrent Collaboration with two files

Now we assume there are file F_1 and F_2 in the system. The system will form sub-systems S_1 and S_2.

	File 1	**File 2**
S_1	**Downloading**	**Downloading**
S_2	**Finished**	**Downloading**

Table 3 two states of system

The life of a peer in the system is illustrated as below:

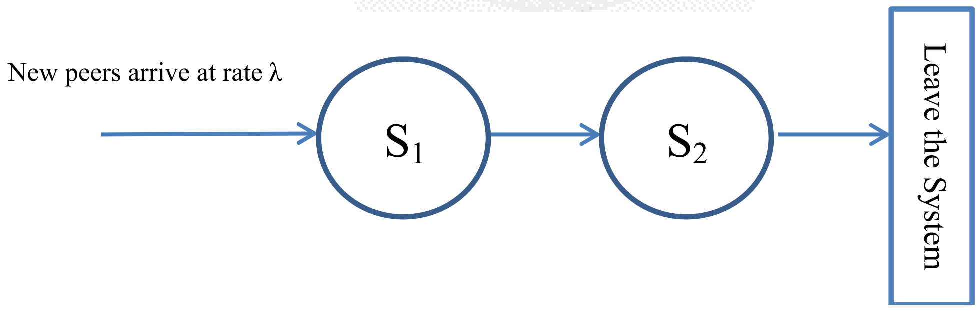

Figure 1 system of 2 files

Now we can list the evolution of number of peers in system S_1 as,

$$\frac{ds_1(t)}{dt} = \lambda - \frac{\eta\mu R\frac{s_1(t)}{2} + (1-R)\mu\frac{s_1(t)}{2}}{F_1}, \qquad (5)$$

To study to performance under steady-state, we let

$$\frac{ds_1(t)}{dt} = 0.$$

From equation (5) we get,

$$0 = \lambda - \frac{\eta\mu R\frac{\overline{s_1}}{2} + (1-R)\mu\frac{\overline{s_1}}{2}}{F_1}, \qquad (6)$$

We get the number of peers in S_1 under steady-state as $\overline{s_1} = \frac{2\lambda F_1}{\mu(\eta R+1-R)}$. Then, we apply the Little's Law [4][4][4][4]. We can get the average time for a peer stay in system S_1 under steady-state T_1.

$$T_1 = \frac{2F_1}{\mu(\eta R + 1 - R)}, \qquad (7)$$

After calculate T_1, we are going to calculate the part of F_2 download during S_1. We believe because of the TFT mechanism the peers in S_1 will only get pieces of F_2 from *Optimistic unchoking* coming from peers in other sub-systems. So the size of file that F_2 being downloaded during S_1 is

$$B_{21} = T_1(1-R)\mu\frac{\left[\frac{\overline{S_1}}{2} + \overline{S_2}\right]\frac{\overline{S_1}}{\overline{S_1}+\overline{S_2}}}{\overline{S_1}}, \qquad (8)$$

Then we move on to sub-system S_2, the evolution of number of peers in S_2 is as follows,

$$\frac{ds_2(t)}{dt} = \lambda - \frac{\eta\mu R S_2(t) + (1-R)\mu\left[\frac{S_1(t)}{2} + S_2(t)\right]}{F_2 - B_{21}}, \qquad (9)$$

As same as what we do in S_1, we let

$$\frac{ds_2(t)}{dt} = 0$$

From (9) we get,

$$0 = \lambda - \frac{\eta\mu R\bar{S_2} + (1-R)\mu\left[\frac{\bar{S_1}}{2} + \bar{S_2}\right]}{F_2 - B_{21}}, \qquad (10)$$

Then, we can calculate the number of peers in S_2 under steady-state $\bar{s_2}$. Then, with the value of $\bar{s_2}$, we can calculate the average time of peers staying in the sub-system S_2 T_2. The total time a peer spends in the system will be $T_1 + T_2$. But unfortunately, even with only n=2, we found the equation became too complicated and difficult to solve.

Even after so many abstractions of network and making some unrealistic assumptions, we still failed to find a valuable solution or an equation that can present the performance of MFMT under steady-state. Because we find that it is hard to capture the performance of MFMT with mathematic modeling, so we decided to change our approach by using a simulation experiment in next chapter. We wish through simulation experiment to discover the attribute that affects the performance of MFMT download.

3.3 Multiple-Files-Single-Torrent

In this chapter we are going to discuss the case of MFST. Actually on paper MFST is almost the same as MFMT, with only one difference: These multiple files distributed across different Torrents in MFMT are now all bundled into a single torrent in MFST. It is quite hard to distinguish their difference through mathematic modeling. In facts, its depends on the way the BitTorrent client actually handles the Multiple-file torrent MFMT and MFST could be the same. As mentioned before Tian et al. [8] do use the same model on them in his work.

Despite using the same model on MFMT and MFST, they still believe that there may be differences between these two different download methods. The difference they mentioned in their works is the concept of file correlation, which we have mentioned before. Another attribute that we think that may affect the difference between these two download methods are the number of *unchoking slots* used.

In most practical BitTorrent clients, the number of allowed *connections* and *choking slots* are restricted within each task. Usually one task is added for every torrent that BitTorrent clients download no matter how many files are within the torrent. So, the difference between MFMT and MFST may occur here. Take for example; downloading the same three files A, B, C and *n* unchoking slots are used for every task in the BitTorrent Clients. If these 3 files are downloading through three separate torrents, a total of *3n* unchoking slots are going to be used; but only *n* unchoking slots are used when these three files are bundled intoa single torrent.

Fan et al. [10] have studied the influence of number of *unchoking slots* to the download performance in BitTorrent. In his model he finds that the ratio between *optimistic unchoking slots* and *regular unchoking* will affect the download performance: the higher the ratio of o*ptimistic unchoking*, the higher the download performance. It is very obvious because the unbiased *Optimistic unchoking slot* compared to the biased *regular unchoking slot* does not have the bandwidth lost due to playing tit-for-tat strategy. Most of the works including [4, 8, 9, 17] have all made the similar assumption that *optimistic unchoking slot* does outperform *regular unchoking slot* in sharing efficiency.

The ratio between two different *unchoking slots* do affect the performance of the BitTorrent system, but if the ratio is maintained, no matter the number of *unchoking slots* are used the performance of system will stay the same [10]. It is quite easy to understand because we usually assume there are a sufficient number of peers in the system and there are always pieces to download if the bandwidth is allowed. But in the real world there may be chances that even though we have enough unoccupied bandwidth, there are no desired pieces to download.

As to our knowledge of most of the commonly used BitTorrent clients, the default number of *optimistic unchoking slots* and *regular unchoking slots* are 1 and 4 respectively. They are set to the number not according to any theory or proof but by experience. We wonder if this setting by past experience may not work that well when considering the existence of multiple-file torrents.

Unfortunately, we cannot theoretically prove that if we extend the number of unchoking slots used according to the number of files in the torrent, that can affect the performance of download. As a result, we decided to examine this situation by running some simulation experiments in the next chapter.

Chapter 4

Experiment Result

In the previous chapter we theoretically examine some different multiple-file download approaches of BitTorrent. We have made some assumptions but there are still many cases that we have failed to analyze. In this chapter, we decided to study them through a series of simulation experiments.

The Simulator Used

The simulator we used is General Peer-to-Peer Simulator (GPS) [13]. GPS is an event-driven p2p simulator which simulates every messages passing between peers. The reason we choose GPS as our simulator is because it already has a built-in BitTorrent protocol and also support for multiple-file downloads.

GPS adopt the GT-ITM [25] as a tool to simulate the topology of a real network. The bandwidth is also constrained according to the network topology. In the network topology that GPS used there are two kinds of nodes *transit node* and *stub node*. Transit nodes are the backbone of the network and stub nodes are the edge of network or the place where peers actually exist. These two different nodes form three kinds of different connections. These connections are the connection between two transits nodes (TT), connection between two stub nodes (SS) and the connection between transit node and stub node (TS).

The bandwidths are allocated according to these connections. The propagation delays on the connections are also considered. All of the above features make GPS closer to a real network environment.

The experiments in GPS are configured by two text-based setting files; document file and event file. Document file defines the size and amount of the download files in the system. Event file defines event happened in the system.

In the following experiment 1 and 2, we are using the default setting of GPS for the bandwidths and delays on different connections. Bandwidth between two transit nodes is 1000Mbps with a 5ms of delay. Bandwidth between transit node and sub node is 100Mbps with a 10ms delay. Bandwidth between two stub nodes is 10Mbps with 30ms delay.

Bandwidth Transit-Transit	**1000 Mbps**
Bandwidth Transit-Sub	**100 Mbps**
Bandwidth Sub-Sub	**10 Mbps**
Delay Transit-Transit	**5 ms**
Delay Transit-Sub	**10 ms**
Delay Sub-Sub	**30 ms**

Table 3 Network setting of experiments 1 and 2

4.1 Experiment 1

In our first experiment, we compare the performance of three different kinds of multiple-file download approaches in BitTorrent. MFMT, MFST, and MFST with extended *unchoking slots*. For each of the three different approaches, 3 sets of files are used as download files; one single 500MB file, two 250MB files and five 100MB files. The number of unchoking slots used is the default setting of most practical BitTorrent clients: one optimistic unchoking slot and four regular unchoking slots, and the numbers of both kinds of unchoking slots are multiplied by 2 in the scenario of extended 2, and multiplied by 5 in the scenario of extended 5.

Number of files	**Single file size**
1	**500 Mb**
2	**250 Mb x 2**
5	**100 Mb x 5**

Table 4 the size of download files in experiment 1

In figure 2, the average download times of these different approaches are shown.

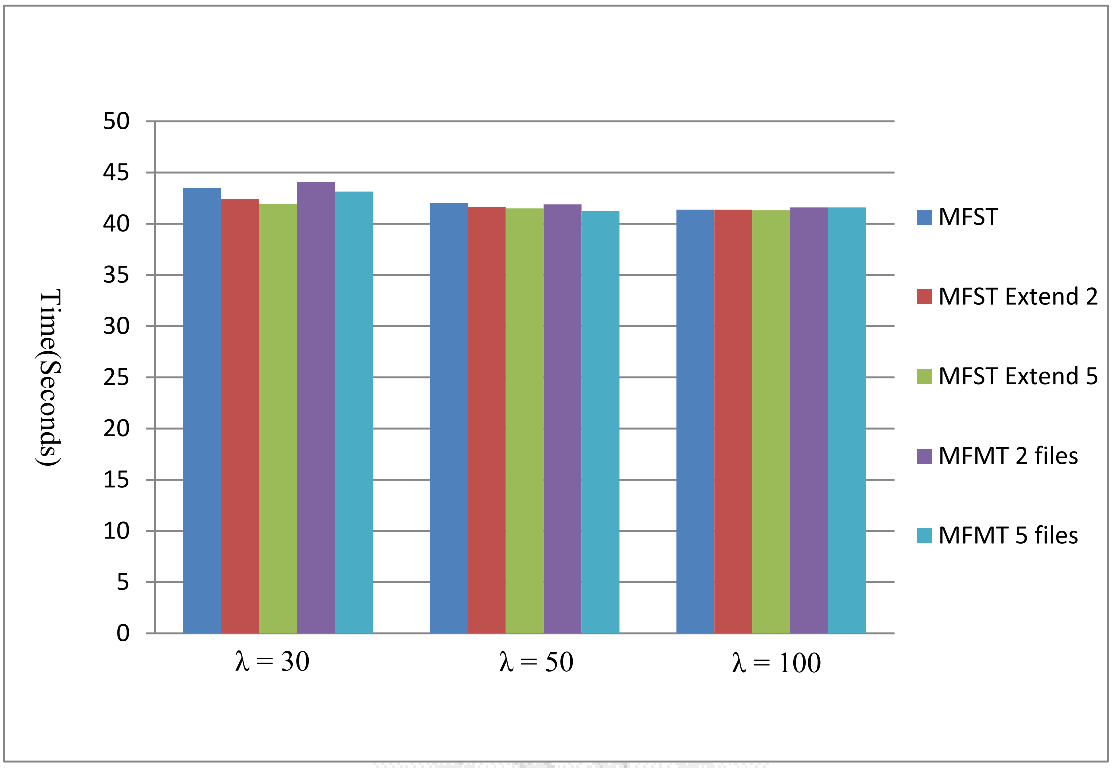

Figure 2 the average download times in experiment 1

In these first experiments, two hundred peers will enter the system at a constant rate (30, 50 and 100) and start to download all the available files in the system. Five different download approaches with 3 different entry rates are simulated. The five different download approaches are MFST, MFST extend 2 (which means the number of unchoking slots are 2 times the default setting), MFST extend 5 (which means the number of unchoking slots are 5 times the default setting); MFMT 2 files are downloading 2 files from 2 separate torrents and MFMT 5 files are downloading 5 files from 5 separate torrents. The total sizes of the download files in all experiments are 500 Mb.

We can see that, as mentioned in the previous chapter, the entry rate of peers does not affect the download times of peers. Also, all five different download scenarios have the similar average downloading times. This is very reasonable because of the high bandwidth utilization in our experiments, especially when the sizes of files are unique in each scenario.

4.2 Experiment 2

In the second experiment we examine the download performance of MFMT under the condition when the sizes of files are different. Figure 3 shows the average download time of experiment 2.

Number of files	**files sizes**
1	**500 Mb**
2	**300 Mb, 200 Mb**
5	**150Mb, 125Mb, 100Mb, 75Mb, 50Mb**

Table 5 the size of download files in experiment 2

In this experiment, the total sizes of all download files are the same as experiment 1, a total of 500 Mb. However, the difference is that sizes of files are no longer unique in the 2 and 5 files scenarios.

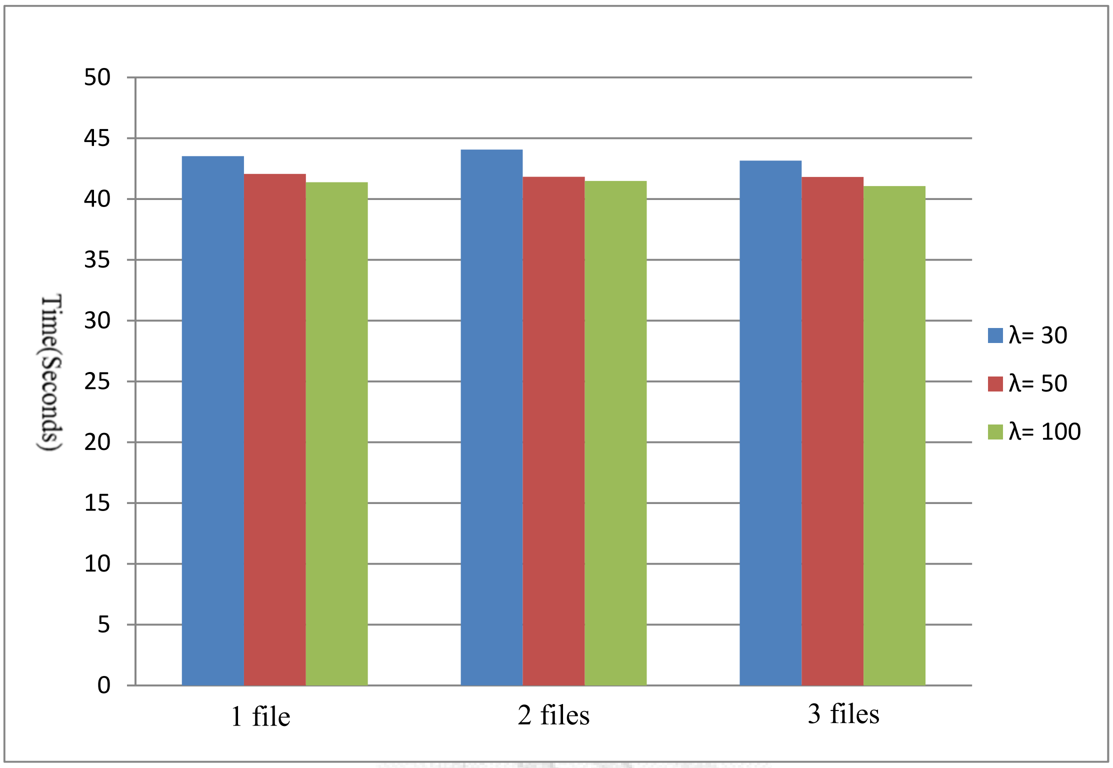

Figure 3 the average download times in experiment 2

From the result shown in figure 3, we can see that even when the size of files are not unique, the average download times of MFMT are still very similar when the total sizes of all download files are the same. The average download time of multiple-file download does not affect by the individual sizes of download files. This is also very reasonable when we consider that the simulation environment high bandwidth utilization combined with almost no waiting times due to the fact that we let all three swarms have the same set of peers.

4.3 Experiment 3

In this experiment we rerun the same experiments that we have done in experiment

1 and experiment 2 but under a slower network environment. The delays between different connections remain the same but bandwidths are adjusted. Bandwidth between two transit nodes is reduced to 500Mbps. Bandwidth between transit node and sub node is reduced to 50Mbps. Bandwidth between two stubs nodes is reduced to 5Mbps.

Bandwidth Transit-Transit	**500 Mbps**
Bandwidth Transit-Sub	**50 Mbps**
Bandwidth Sub-Sub	**5 Mbps**

Table 6 Network setting of experiment 3

Rerun of Experiment 1

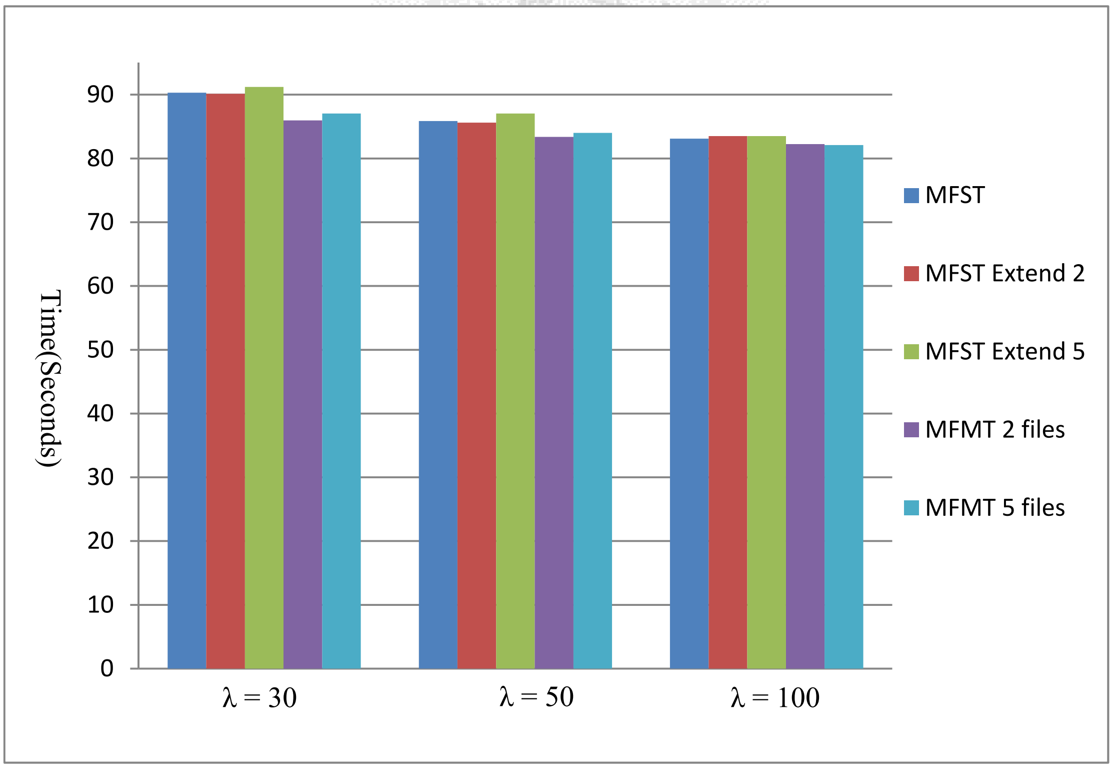

Figure 4 the average download times of experiment 1 rerun

From figure 4 we can see the performance of different download methods are actually very similar. The result strengthens the conclusion we made in the previous experiment: that when network utilization is high, the average download times of peers is not affected by the way they choose to download the files.

Aside from the conclusion we already know, there is something interesting we observed from our results. Even though average download times across different download methods are similar, but there are notable differences between the performance between λ=30 and λ= 100 under low bandwidth environment. The result is very easy to understand because we limit our network bandwidth capacities. So, under a low bandwidth environment with shorter average entry time, the upload bandwidth of peers provided reached the limit capacities of backbone network, resulting longer average download times.

Rerun of Experiment 2

Just like the previous rerun of experiment 1, we observed the same result as experiment 2 at this rerun. Different download methods actually do not influence the average download times. We also find the trend that when the average arrival times are lower the average download times are shorter. This is due to the fact that faster arrival of peers may cause backbone network capacities be occupied.

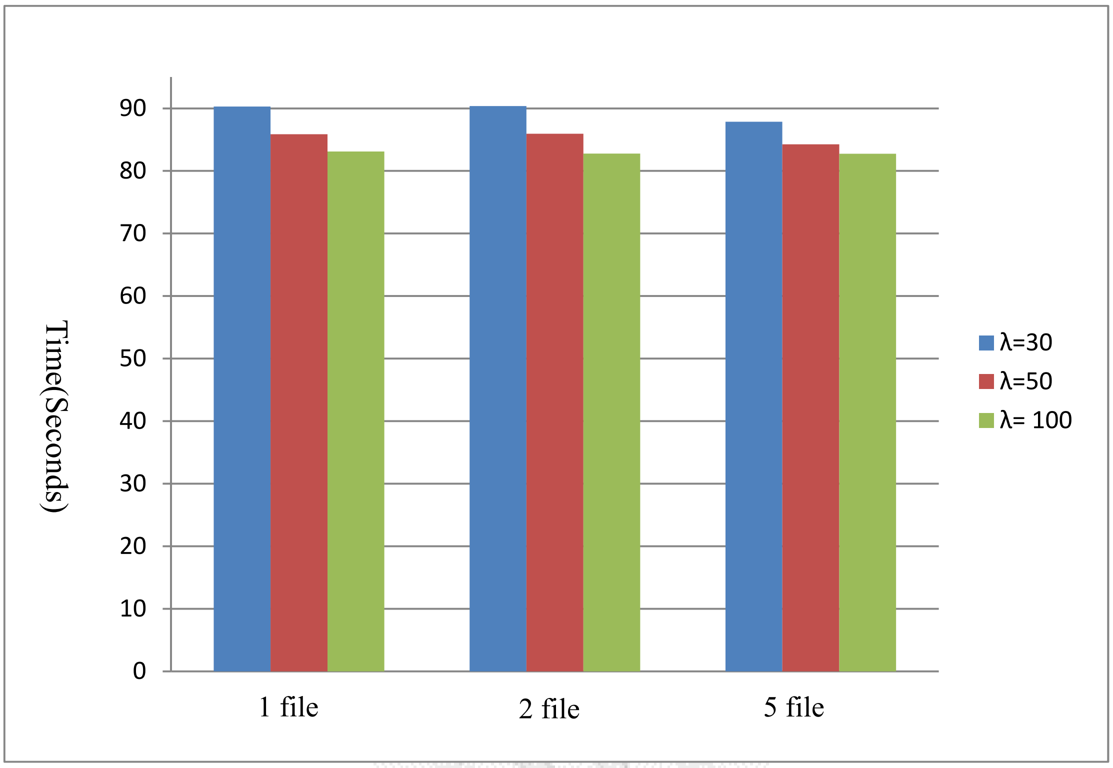

Figure 5 the average download times of experiment 2 rerun

In this chapter we mainly run three sets of experiments. The first one is to compare the influence of different download approaches to the average download time. The second one is to examine the situation in which sizes of files are not unique when using MFMT download approaches. The last one is a rerun of previous two experiments under a low bandwidth environment. The result showed no matter under high or low bandwidth conditions, the download approaches used or the size difference between download files does not affect the average download speed. We also find that when network bandwidth capacities are low, entry rates actually may affect the download performance, because too many peers may occupy the bandwidth, causing longer average download times.

Chapter 5

Conclusions

In this paper, we first introduced the various different methods of multiple-file downloads in BitTorrent. Then, we classified and introduced the differences between these approaches. After, we tried to study the performance of different multiple-file download approaches under steady-state based on the previous research from others and some new models of our own.

Unfortunately, our attempt to model the downloading time of MFMT download failed. We tried to study the performance of MFMT with simulation experiments. The results show that MFMT with Inter-torrent collaboration, MFMT without Inter-torrent collaboration and MFST have similar performances.

We also extended the number of *unchoking slots* in MFST when there are multiple files in the single torrent. The results show the same as other previous research on a single file system, that the number of *unchoking slots* does not affect the download performance under the steady-state.

Future improvement

We tried to use the commonly used Fluid model to analyze the different approaches of multiple-file download in BitTorrent. Unfortunately, even with so much abstraction of network details, we still failed to find a meaningful result to describe the performance of MFMT. We hope in the future we can develop a new model that is able to better describe the performance of multiple-file download in BitTorrent under various Network conditions.

We have run a series of experiments by using GPS simulators. Even though simulation is a good way to capture the performances of BitTorrent under certain network conditions, we can still consider doing simulation in a real-world network environment. Or, we can even try to collect data and tracker trace from real-world BitTorrent clients and tracker, resulting in a more diverse and complete research on multiple-file download in BitTorrent.

Bibliography

[1] *BitTorrent*. Available: http://www.bittorrent.com/

[2] B. Cohen, "Incentives Build Robustness in BitTorrent," presented at the 1st Workshop on Economics of Peer-to-Peer Systems, 2003.

[3] sandvine, "Global Internet Phenomena Report," Fall 2011.

[4] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," presented at the Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, Portland, Oregon, USA, 2004.

[5] B. Fan, J. C. S. Lui, and D.-M. Chiu, "The design trade-offs of BitTorrent-like file sharing protocols," *IEEE/ACM Trans. Netw.,* vol. 17, pp. 365-376, 2009.

[6] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1-12.

[7] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of BitTorrent-like systems," presented at the Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, Berkeley, CA, 2005.

[8] T. Ye, W. Di, and N. Kam-Wing, "Analyzing Multiple File Downloading in BitTorrent," in *Parallel Processing, 2006. ICPP 2006. International Conference on*, 2006, pp. 297-306.

[9] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," *Selected Areas in Communications, IEEE Journal on,* vol. 25, pp. 155-169, 2007.

[10] F. Bin, J. C. S. Lui, and C. Dah-Ming, "The Design Trade-Offs of BitTorrent-Like File Sharing Protocols," *Networking, IEEE/ACM Transactions on,* vol. 17, pp. 365-376, 2009.

[11] N. Laoutaris, D. Carra, and P. Michiardi, "Uplink allocation beyond choke/unchoke: or how to divide and conquer best," presented at the Proceedings of the 2008 ACM CoNEXT Conference, Madrid, Spain, 2008.

[12] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," presented at the PAM, 2004.

[13] W. Yang and N. Abu-Ghazaleh, "GPS: a general peer-to-peer simulator and its

use for modeling BitTorrent," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, 2005, pp. 425-432.

[14] *Bittorrent Protocol Specification v1.0*. Available: http://wiki.theory.org/BitTorrentSpecification

[15] *The Pirate Bay*. Available: http://thepiratebay.org/

[16] W.-C. Liao, F. Papadopoulos, and K. Psounis, "Performance analysis of BitTorrent-like systems with heterogeneous users," *Perform. Eval.,* vol. 64, pp. 876-891, 2007.

[17] A. L. H. Chow, L. Golubchik, and V. Misra, "BitTorrent: An Extensible Heterogeneous Model," in *INFOCOM 2009, IEEE*, 2009, pp. 585-593.

[18] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," presented at the Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeriro, Brazil, 2006.

[19] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis," presented at the 4TH INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS), 2005.

[20] Z. Chao, P. Dhungel, W. Di, L. Zhengye, and K. W. Ross, "BitTorrent Darknets," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1-9.

[21] G. Neglia, G. Reina, Z. Honggang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 2216-2224.

[22] Y. Yang, A. L. H. Chow, and L. Golubchik, "Multi-torrent: a performance study and applications," *Int. J. Adv. Media Commun.,* vol. 4, pp. 31-58, 2010.

[23] J. Han, T. Chung, S. Kim, H. Kim, T. T. Kwon, and Y. Choi, "An Empirical Study on Content Bundling in BitTorrent Swarming System," *CoRR,* vol. abs/1008.2574, 2010.

[24] D. S. Menasche, A. A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani, "Content availability and bundling in swarming systems," presented at the Proceedings of the 5th international conference on Emerging networking experiments and technologies, Rome, Italy, 2009.

[25] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, 1996, pp. 594-602 vol.2.