The Hong Kong Polytechnic University

Department of Electronic & Information Engineering

MODELING, ANALYZING AND IMPROVING THE PERFORMANCE OF BITTORRENT SWARMING SYSTEMS

QINGCHAO CAI

A thesis submitted in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

September 2012

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

Qingchao Cai (Name of student)

I dedicate this dissertation to my parents, sisters, girl friend, family, professor and friends.

Abstract

BitTorrent is one of the most popular peer-to-peer content distribution systems, and plays a dominant role with respect to the Internet traffic. Although BitTorrent is very effective in terms of bandwidth utilization, it is confronted with a serious problem that in many BitTorrent swarms, peers cannot complete the download due to the lack of some content blocks. Therefore, it is very important to find solutions to this problem, which we call *content availability*, as they can significantly enhance the service capability and performance of BitTorrent swarms.

This work aims to develop an insightful understanding to the performance of BitTorrent swarming systems, and explore how it can be improved, with a special focus on content availability. In this study, we first perform a comprehensive study on the modeling and analysis of BitTorrent swarms. We derive the closed-form expressions for the performance metrics of BitTorrent swarms related to content availability, and investigate the influence of bundling on content availability. It is shown that bundling could greatly improve the availability of content, and that in a bundled swarm, peers could complete the download earlier than they would do in the individual swarm, given an appropriate number of files are bundled. In addition, the altruistic behavior of peers is also studied. We present an analysis on how peers' altruistic behavior affects the length of the residual active period after the leave of the publisher, and quantify the impact of bundling on the residual active period in the presence of peers' altruistic behavior.

Next, we carry out an in-depth investigation on the feasibility of using network coding to ameliorate content availability of BitTorrent swarms. We first present a mathematical analysis on the potential improvement in the content availability and bandwidth utilization induced by two existing network coding schemes. The analysis reveals that network coding has a large potential to improve content availability, but both of the existing two schemes are not feasible as they either incur a very high coding complexity and disk operation overhead or cannot effectively leverage the potential of improving content availability. In this regard, a simple sparse network coding scheme is proposed, which addresses both the drawbacks in the existing schemes, and a new block scheduling algorithm is also developed in order to accommodate the proposed coding scheme into BitTorrent. The extensive simulation results demonstrate the effectiveness of the proposed coding scheme in terms of improving content availability.

Finally, as motivated by the recent development of private BitTorrent communities, we conduct a detailed survey on one of the largest private BitTorrent communities, CHDBits. First, we characterize torrents from the perspectives of age, size, popularity and average user download rate, and then profile the different aspects of CHDBits users, e.g., diurnal access pattern, user traffic, seeding and leeching time. We also develop an in-depth understanding to how CHDBits users participate in downloading and uploading. The survey results suggest some new findings with regard to user behavior: low bandwidth users are more likely to participate in torrents with a smaller content size or a higher popularity, and compared with low bandwidth users, high bandwidth users tend to participate in more torrents, but spend less time in seeding.

List of Publications

[1] Qingchao Cai and Kwok-Tung Lo. Two blocks are enough: on the feasibility of using network coding to ameliorate the content availability of BitTorrent swarms. *IEEE Transactions on Parallel and Distributed Systems*, accepted to appear.

[2] Qingchao Cai and Kwok-Tung Lo. An analysis of user behavior in a private BitTorrent community. *International Journal of Communication Systems*, accepted to appear.

[3] Qingchao Cai and Kwok-Tung Lo. Modeling and analysis of content availability and bundling in BitTorrent-like file swarming systems. Under submission.

[4] Qingchao Cai and Kwok-Tung Lo. A detailed survey on a large private BitTorrent community. Under submission.

[5] Qingchao Cai and Kwok-Tung Lo. Incentivize BitTorrent peers to simultaneously upload to more neighbors. Under submission.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Kwok-Tung Lo, for his continuous support during my Ph.D. study, for his patience, encouragement and advice, which enable me to proceed through the doctoral program. His guidance helped me throughout my Ph.D. study and in writing of this dissertation, and will continue to help me in my future research.

I wish to thank my parents and sisters for their understanding, endless patience and unconditional support. Their love is one of my greatest fortune. I also wish to express deep gratitude to my girl friend for her consistent support, uncomplaining waiting, and encouragement when it was most required, all of which have been and continue to be my driving force.

My sincere thanks are due to Dr. Xue-Jie Zhang, the director of my master's thesis, for his substantial help in my research and study during master's program at Yunnan University, and encouraging me to pursue a Ph.D. degree.

I would like to thank my colleagues at Hong Kong Polytechnic University for the nice time that we had in the last three years. I also wish to thank my friends and relatives for their help and moral support.

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1
Introduction

1.1 Background

Content distribution is one of the most important and popular classes of Internet applications. The traditional client-server based approach for content distribution is known to be unable to scale with the client demand as the quality of service provided by the server dramatically degrades with the increase of demand. In addition, the high cost in the server also poses an obstacle to the individuals who intend to disseminate their own content using this approach.

The emergence of peer-to-peer paradigm provides an alternative way for content distribution, and BitTorrent [27] is the most widely used peer-to-peer application for content distribution and plays an important role in terms of the traffic generated [8, 88]. As reported in [8], there are hundreds of millions BitTorrent users, accounting for a significant amount of today's Internet traffic.

The key idea of BitTorrent is to leverage the outgoing bandwidth of participants, which significantly reduces the load and cost on the content publisher and enhances the scalability, as more participants also imply an increase in the overall available outgoing bandwidth. By splitting a file into many blocks, BitTorrent enables peers with an arbitrary number of blocks to exchange blocks with other peers, provided that peers involved in exchange have different blocks. Therefore, the bandwidth of participating peers can be effectively utilized in BitTorrent swarms[1]. BitTorrent is also highly scalable as more participating peers lead to an increase in the overall bandwidth resources, and thus accelerate the process of file distribution.

[1] A BitTorrent swarm is composed of all the peers that participate in the distribution of the content.

CHAPTER 1

Although the effectiveness of BitTorrent in utilizing the outgoing bandwidth of participating peers has been shown in many studies by the means of quantitative analysis [98, 83, 69], and measurement [54, 13], BitTorrent is confronted with a serious problem that significantly influences user-perceived quality of service. This problem is that, as identified in [70], many BitTorrent swarms suffer a high degree of *content unavailability*[1], and peers in these swarms cannot complete the download as a consequence of lost blocks. Therefore, it is of significant importance to find potential solutions to enhance the availability of content in BitTorrent swarms, which would in turn improve the service capability and performance of BitTorrent swarms.

Content bundling is a potential way to improve the availability of content in BitTorrent swarms. As the name implies, content bundling is a content publish strategy that bundles several similar contents, e.g., a TV season, a series of movies acted by the same person, and publish them in a single BitTorrent swarm. This strategy has been now widely adopted by many BitTorrent users in publishing content, and can be observed in a great number of BitTorrent swarms [42, 43]. Moreover, in bundled swarms, many peers tend to download almost the whole bundle, even if they are given the freedom to select which files to download[42].

Although content bundling might be originally developed to facilitate the collection of contents of interest for BitTorrent users, it in fact has the ability to enhance the content availability. Specifically, in a swarm where several related files (contents) are bundled together, peers interested in each file might come, which means an increase in peer arrival rate, and peers may have to stay longer in the swarm to complete the download as the bundle is of a larger size than the individual content. Therefore, content bundling can substantially increase the number of participating peers, which in

[1]The content is available in a BitTorrent swarm when all the blocks of the target file (content) are available, and if there is at least one missing block, we say that the content is not available in the swarm, as the remaining blocks cannot reconstruct the original content.

turn reduces the possibility that chunks, i.e., blocks[1], are lost when there is no seeder[2] in the swarm, thereby enhancing the availability of content.

It is thus interesting to quantify how much bundling could enhance the content availability of BitTorrent swarms. Menasche et al. [70] studied this problem by modeling BitTorrent swarms as $M/G/\infty$ queues. However, most results of [70] were derived based on the assumption that the population threshold, the minimum number of leechers [3] to have all chunks, is equal to one, which is not realistic since a peer could complete the download only when there is at least one seeder or two leechers. Therefore, it is necessary to investigate this problem in a more realistic scenario in which the population threshold is larger than one, which corresponds to the first contribution of this work.

Linear network coding is another possible way to enhance the content availability of BitTorrent swarms. When using network coding, the blocks transferred among peers are linear combinations of the original blocks, and the content can thus be recovered from the encoded blocks as long as the rank of the coefficient vectors of these encoded blocks is equal to the number of original blocks by solving a system of linear equations. However, accompanying the improvement in the content availability is the cost of additional disk read/write and computation involved in the encoding and decoding process. This motivates us to investigate the feasibility of using linear network coding to enhance the availability of content from the perspectives of performance enhancement and the cost of computation and disk read/write incurred by linear network coding, which corresponds to the second contribution of this work.

Content being unavailable only occurs in the swarm where there are no seeders, and if there is at least one seeder in the swarm, then the availability of content can be guaranteed, as a seeder has all blocks. However, BitTorrent does not provide incentives

[1] Throughout the dissertation, we use these two words interchangeably.

[2] A seeder is a peer who has all the file blocks, and a peer becomes a seeder when it completes the download.

[3] A leecher is a peer who has not completed the download.

to encourage peers to stay at the swarm as seeders after completing the download. As a result, due to the nature of unwillingness to contribute without being rewarded, many peers immediately leave the swarm at the time when the download is completed, and the content thus may become unavailable quickly after the leave of the content publisher which initiates the swarm. Therefore, the mechanism that encourages or forces peers continue to stay at the swarm after completing the download is a very effective way to promote the availability of content.

Such mechanisms have been implemented in private BitTorrent communities by the means of only allowing the specific peers to download the content and bringing in a stringent restriction on how much data a peer should upload to others. A private BitTorrent community is mainly composed of users who have some common interest, and in general, only the contents of interest, e.g., high definition movies, animation, music, TV episodes, are allowed to be published and disseminated in the community. In a private BitTorrent community, only the registered users can browse and download the torrent files of this community. Users must maintain a minimum share ratio, namely the ratio of uploaded data volume to downloaded data volume, to prevent from being warned or banned, which is known as SRE (Share Ratio Enforcement). Due to the existence of peer admission policy and SRE, peers in private BitTorrent communities can enjoy a high download rate, and the content of many torrents remains available even after a long time since the release of the torrents [18]. For these reasons, private BitTorrent communities have experienced a rapid development in recent years. According to [102], there have been over 800 active private BitTorrent communities. Motivated by the recent significant development of private BitTorrent communities, we conduct a detailed survey on one of the most representative private BitTorrent communities, which corresponds to the third contribution of this work.

1.2 Contributions and previous work

This work aims to develop an insightful understanding to the performance of BitTorrent swarming systems, and explore how it can be improved, with a special focus on content availability.

1.2.1 Content availability and bundling in BitTorrent swarms

Menasche et al.[71] analyzed chunk availability in BitTorrent swarms by modeling download process as a tandem Jackson network. Susitaival et al.[96] presented an insightful view of content availability by considering the periods when content was available as busy periods of an M/G/∞ queue. This view of content availability is also adopted in [70]. The authors of [70] further quantified content availability and the impact of bundling on availability and average sojourn time. However, in [70], most of the results were derived based on the assumption that the population threshold is equal to one, and only the result of the active period was provided for the cases in which the population threshold is larger than one.

There also existed some studies on content bundling in BitTorrent network. [41] argued that many contents can be bundled together due to the high similarity among them. It was further pointed out in [43] that content bundling is now very common in BitTorrent swarms, and the impact of content bundling on the performance metrics of BitTorrent swarms was empirically studied in [42]. [72] investigated the bundling strategies for publishers, and [55], on the other hand, carried out an analysis on the file selection strategies for peers in bundled swarms. In addition, a dynamic bundling system which requires peers to download the files in additional to those of their interest to enhance the content availability was presented in [103].

In this study, we first perform a comprehensive study on the modeling and analysis of BitTorrent swarms. We derive the closed-form expressions for the performance metrics of BitTorrent swarms related to content availability, and investigate the influence

of bundling on content availability. It is shown that bundling could greatly improve the availability of content, and that in a bundled swarm, peers could complete the download earlier than they would do in the individual swarm, given an appropriate number of files are bundled. In addition, the altruistic behavior of peers is also studied. We present an analysis on how peers' altruistic behavior affects the length of the residual active period after the leave of the publisher, and quantify the impact of bundling on the residual active period in the presence of peers' altruistic behavior. This part of our work extends [70] by re-deriving the performance metrics related to content availability and quantifying the impact of bundling based on the assumption that the population threshold is larger than one, and thus complements [70].

1.2.2 The applications of network coding to peer-to-peer networks

Network coding was originally proposed in information theory [10], and thereafter it was introduced to peer-to-peer networks [25] [49]. Since then, network coding has gradually demonstrated its power in improving the overall performance of peer-to-peer networks. In [38], a file sharing protocol called Avalanche was proposed, which uses random linear network coding to accelerate the download process and facilitate the block scheduling among neighbors. In [100], Wang et al. proposed a live peer-to-peer streaming protocol, R^2, which also implemented random network coding to enable the coded blocks to be randomly push between neighbors without block availability information. There are also some commercial peer-to-peer streaming softwares [65] using network coding for effective content distribution.

The random network coding adopted in Avalanche [38] can also improve content availability. With the random linear network coding adopted in [38], each coded block is a linear combination of all plain blocks of the file, and the original file can be recovered as long as the dimension of the space spanned by the coding coefficient vectors of the coded blocks in the swarm is equal to the number of plain blocks, which occurs with high probability even the number of coded vectors is exactly the same as that of original

blocks. In addition, a sparse form of random linear network coding is proposed in [76], in which only two random plain blocks are used to generate new blocks.

In our second part of our work, we carry out an in-depth investigation on the feasibility of using network coding to ameliorate content availability of BitTorrent swarms. We first present a mathematical analysis on the potential improvement in the content availability and bandwidth utilization induced by two existing network coding schemes [38, 76]. The analysis reveals that network coding has a large potential to improve content availability. However, we also show that both of the existing two schemes are not feasible as they either incur a very high coding complexity and disk operation overhead or cannot effectively leverage the potential of improving content availability due to the slow generation rate of new blocks. In this regard, a simple sparse network coding scheme is proposed, which addresses both the drawbacks in the existing schemes, and a new block scheduling algorithm is also developed in order to accommodate the proposed coding scheme into BitTorrent. The extensive simulation results demonstrate the effectiveness of the proposed coding scheme in terms of improving content availability.

1.2.3 Private BitTorrent community

Due to its increasing prevalence, private BitTorrent communities received much attention recently. A couple of studies were carried out to characterize these communities. In the recent work [102, 22, 73, 46], it was found that in private BitTorrent communities, the ratio of seeders to leechers is much higher than that in public communities. Compared with users in public communities, users in private BitTorrent communities tend to perceive a better download performance, and meanwhile maintain a higher share ratio. [64] and [22] also explored the effectiveness of share ratio enforcement (SRE), a policy commonly adopted in private BitTorrent communities, in improving the user share ratio. However, it was stated in [47] that the existence of SRE may result in users having to seed for an extremely long time in order to keep their share ratio upon a specific level, and different strategies were proposed in [46] for users of private

BitTorrent communities to effectively improve their share ratios. An economic explanation to private BitTorrent communities was presented in [50]. In addition, targeting on the phenomenon that some malicious users may cheat the tracker by reporting a modified amount of traffic, Liu et al. [64] further presented an upload entropy scheme for deterring collusion among users. However, in these studies, only limited information about user traffic was presented, and the other two important respects of user behavior: seeding and leeching time, and download history, were not investigated. In addition, in the existing literature, how user behavior is influenced by user age and bandwidth was still unknown, as well as the impact of the content size and popularity of torrents on user behavior.

As motivated by the recent development of private BitTorrent communities, we conduct a detailed survey on one of the largest private BitTorrent communities, CHDBits [9], in the last part of our work. First, we characterize torrents from the perspectives of age, size, popularity and average user download rate, and then profile the different aspects of CHDBits users, e.g., diurnal access pattern, user traffic, seeding and leeching time. We also develop an in-depth understanding how CHDBits users participate in downloading and uploading, and how user participation is affected by the various factors, such as user bandwidth, and the content size and popularity of torrents, thereby bridging the gaps in the existing literature [102, 22, 73, 46]. The survey results reveal some new findings with regard to user behavior: low bandwidth users are more likely to participate in torrents with a smaller content size or a higher popularity, and compared with low bandwidth users, high bandwidth users tend to participate in more torrents, but spend less time in seeding.

1.3 Thesis structure

The remainder of this thesis is organized as follows. We present a comprehensive review on the literature related to our thesis in Chapter 2. Chapter 3 studies the

content availability of BitTorrent swarms, and how it can be improved by bundling and peers' altruistic behavior. In Chapter 4, we explore the feasibility of using network coding to ameliorate the content availability of BitTorrent swarms. In Chapter 5, we present a detailed survey on a large private BitTorrent community, revealing some new findings with respect to user behavior. Chapter 6 concludes this thesis, and presents some directions for future work.

Chapter 2
Literature Review

In this chapter, a comprehensive review of the studies related to BitTorrent is presented. We first introduce the taxonomy of peer-to-peer networking, and position BitTorrent in the taxonomy. We then investigate the two major classes of BitTorrent study: performance analysis, and protocol design; private BitTorrent community, which is becoming increasingly popular in recent years, is also explored. Finally, we discuss the applications of network coding to peer-to-peer networking.

2.1 Peer-to-peer networking

Peer-to-peer networking is one of the most important computing paradigms in today's Internet. In a peer-to-peer network, each peer is both the provider and consumer of service, which is different from the traditional client/server paradigm in which the service provider and consumer are strictly differentiated, and the system load is distributed among all the peers, which renders peer-to-peer networking much more scalable than client/server paradigm, as more participating peers means more service capacity.

In a peer-to-peer network, an overlay is built on the top of physical network topology. According to the overlay organization, peer-to-peer networks can be divided into two categories: structured peer-to-peer network and unstructured peer-to-peer network. In structured peer-to-peer networks, peers and documents are mapped into a same address space using distributed hash table, and a peer is responsible for the management of information [1] about the documents with the similar address (this is why such networks are "structured"), responds the query with the information about matched documents,

[1] The information generally includes the ip address and port of the peer owning the document, which can be used for the query requester to retrieve the document from this peer. In addition, this information is also likely to include the meta-data of the document.

and forwards the queries for a document to a neighbor which is topologically closer to the peer responsible for the management of the document if no matched document is found. Some representatives of such systems are Chord [93], CAN [84], Pastry [87] and Tapestry [104].

The unstructured peer-to-peer networks can further be divided into three sub-categories according to the management of the overlays. The first one is centralized peer-to-peer networks, which consists of two components: central index server and peers. The central index server maintains the information about the documents of all the peers. A peer who wants to acquire a document, must first ask the central server to get a list of peers having the desired document, and then contact these peers to retrieve the document. Therefore, the central server is vital to the centralized peer-to-peer networks. The most well-known representatives of this category are Napster and BitTorrent [2].

The second one is decentralized peer-to-peer network, in which the system load is uniformly distributed among all the peers. In a decentralized peer-to-peer network, each peer, connects to a subset of other peers, is responsible for management of its own documents, and does not know the information regarding the documents hosted in other peers, which is unlike the structured peer-to-peer networks in which each peer knows the location of the document. Upon receiving a query, a peer compares the query string with its own documents, returns the information about the matched documents, and forwards the query to its neighbors. The two examples of this kind of network are Gnutella [4] and Freenet [3].

The last one is hybrid unstructured peer-to-peer networks, a combination of the centralized and decentralized peer-to-peer networks. In a hybrid peer-to-peer network, peers are divided into two classes: *super peers*, and *normal peers*. The super peers have the information about the documents of its neighboring normal peers, and are responsible for routing the queries. Upon receiving a query, either form its neighboring

normal peers or super peers, a super peer forwards the query to the neighboring normal peers where the matched documents can be found, and to other super peers. When a normal peer receives a query from a neighboring super peer, it only returns the information about the matched documents to this super peer without forwarding to other peers. The determination of super peer can be in various ways, e.g., capacity-based and round robin. Kazaa[5] is a representative of hybrid unstructured peer-to-peer network.

2.2 BitTorrent

2.2.1 BitTorrent protocol

As we have mentioned, the key idea of BitTorrent is to leverage the outgoing bandwidth of participating peers, including both the seeders and leechers. To achieve this goal, the distributed file is split into many blocks with equal size, and each peer is able to send their own blocks to others as long as they have not yet acquired these blocks. Therefore, in BitTorrent swarms, the participating peers can get their outgoing bandwidth effectively utilized after receiving a small number of blocks. In addition, BitTorrent is also highly scalable as more participating peers lead to an increase in the overall bandwidth resources, and their service requests can thus be satisfied with little influence on the perceived service quality.

To initiate the deployment of a BitTorrent swarm, a file with the extension of *.torrent* is necessary. This file contains the meta-data information of the file(s) to be distributed, including the file/directory name, block size, a list of hash values of blocks, path and length of each file in the directory[1], and the announce *url* of the tracker. The *tracker* is another important component for the initiation of a BitTorrent swarm. It records the identification information, i.e., IP address/port pair, for each participating peer, and helps organize and maintain the topology of the BitTorrent swarm. In order

[1]This information is available when multiple files are bundled together.

to join a BitTorrent swarm, one first needs to acquire the corresponding *.torrent* file, which can be done by downloading from a torrent publish site, e.g., The Pirate Bay [7], and then periodically requests the announce *url* of the tracker to obtain a list of current participating peers. After that, it can connect to some other peers and start data transfer.

In general, a BitTorrent peer simultaneously unchokes only a part of neighbors[1], i.e., connected peers, and it is thus necessary for BitTorrent peers to decide which neighbors are to be unchoked. BitTorrent employs a *tit-for-tat reciprocal algorithm* to help make this decision: at the beginning of each unchoking round, with a default duration of 10 seconds, the peer unchokes a specific number of, typically 4, neighbors from which it has downloaded most among all neighbors during the last two rounds.

Once a peer has been unchoked by a neighbor, it first finds the block that is owned by this neighbor and least distributed among all neighbors, and then sends the request for this block to the neighbor, which is known as the *rarest-first block scheduling algorithm.* After receiving the requested block from the neighbor, this peer will send the request for another block to this neighbor to saturate the underlying connection.

Besides the regular unchokes which are determined by the tit-for-tat reciprocal algorithm, each BitTorrent peer unchokes another neighbor, which is called *optimistic unchoke*, in order to find if there is any other neighbor that can provide better download performance, and the selection of optimistic unchoke is performed once every three unchoking rounds. In addition, once a peer has obtained all but the last few blocks, it enters the *endgame* mode, and sends the requests for all missing blocks to all neighbors. In order to save bandwidth, upon receiving a block, a peer in the *endgame* mode will invalidate the requests for this block by sending a *cancel* message to all neighbors.

[1] In the context of BitTorrent, a peer unchoking a neighbor implies that it allows this neighbor to download from itself, and the unchoked neighbors are sometimes simply called the *unchokes*.

2.2.2 Performance study

There have been a great amount of work studying the performance of BitTorrent from various aspects. Generally, these studies can be divided into two classes according to the methodology adopted: mathematical analysis and measurement study.

2.2.2.1 Mathematical models of BitTorrent swarming systems

In the pas decades, a number of studies were carried out to model the BitTorrent systems. The transient characteristics of a simple file swarming system was analyzed in [98, 91] using different methods. Veciana and Yang [98] adopted a branching process in the study, and Simatos et al.[91] presented an urn and ball model in their analysis. The system studied in these two works can be viewed as a simplified version of a BitTorrent swarming system: while both seeders(servers) and leechers can provide service to the leechers in real BitTorrent swarming systems, in the system studied in [98, 91], a peer can only download from the servers, and cannot provide service to others until after becoming the server, i.e., completing the download. Veciana and Yang [98] also studied the steady-state performance of BitTorrent swarms by using a Markovian model.

Motivated by [98], Qiu and Srikant [83] investigated peer evolution and the scalability of BitTorrent swarms by using a simple fluid model, and further indicated that the swarming efficiency of BitTorrent is particularly high. The same fluid model was also used in [82] to validate the existence of the steady-state and global stability of BitTorrent swarming systems. Another fluid model based on stochastic differential equation was presented in [36] to analyze the steady-state behavior of BitTorrent swarming systems. Liu and Chen [66] derived the same result regarding peer evolution as [83] by a statistical model. The evolution of peer in a BitTorrent swarm was also extensively studied in [69, 97]. Massoulie and Vojnovic [69] studied two swarming system scenarios with and without exogenous peers by modeling such systems as coupon replication systems in which users are characterized by their current collection of coupons and

exchange coupons with other users to collect the missed coupons, and it was argued in [69] that the swarming performance of BitTorrent-like file swarming systems does not critically rely on altruistic peer behavior, i.e., peer continuing to serve others after completion of the download, or the block scheduling algorithm such as the built-in rarest-first policy of BitTorrent. Ye et al. [97] derived the distribution of peers in different states with respect to the size of file they have downloaded, and discussed the impact of departure behavior of seeders and leechers on this distribution. Arthur and Panigraphy [12] modeled the BitTorrent swarming system as a graph, and leveraged this model to investigate the swarming efficiency of several block scheduling algorithms.

Menasche et al.[71] analyzed chunk availability in BitTorrent swarms by modeling the BitTorrent swarm as a Jackson network consisting of $n + 1$ queues, where n is the number of chunks. In this Jackson network, a peer with k chunks belongs to the k-th queue, and is routed to $(k + 1)$-th queue when it acquires the next chunk. Susitaival et al.[96] presented an insightful view of content availability by considering the periods when content was available as busy periods of an M/G/∞ queue. This view of content availability is also adopted in [70], and the authors of [70] further quantified content availability and the impact of bundling on availability and average sojourn time. However, in [70], most of results were derived based on the assumption that the population threshold is equal to one, and only the result of the active period was provided for the cases in which the population threshold is larger than one. Our work in chapter 3 extends [70] by re-deriving the performance metrics related to content availability and quantifying the impact of bundling based on the assumption that the population threshold is larger than one, and thus complements [70].

In addition, there also have been some studies aiming to predict the average download performance of BitTorrent swarms. Kumar and Ross [52] used a simple fluid model to derive the minimum file distribution time in scenarios similar to BitTorrent swarm. A simple model is used in [59] to predict the average download time in a heterogeneous

BitTorrent swarm. In [23], It was pointed out that the traditional estimation of average download time of peers in a BitTorrent swarm based on the average service capacity was shown to be inaccurate due to the heterogeneous service capacity of different source peers and the fluctuation in the service capacity of a single source peer.

2.2.2.2 Measurement and simulation studies

There were also many studies analyzing the performance of BitTorrent swarming systems through measurement- and simulation-based methods. On the basis of tracker traces, the evolution of torrents and service quality perceived by peers were investigated in [40, 39, 45], and it was pointed out in [40, 39] that due to the exponential decrease of peer arrival rate, service may soon become unavailable. As in the mathematical study [83], the high bandwidth utilization of BitTorrent peers was also identified in some measurement studies [40, 81, 14]. Dale and Liu [29] presented a microscopic explanation to the high bandwidth utilization of BitTorrent peers by investigating the distribution and evolution of data chunks in the BitTorrent swarm. The phenomenon that peers with similar bandwidth in downloading may cluster together when downloading the same torrent was discussed in [53] and [58]. Neglia et al. explored in [75] the availability of trackers in BitTorrent swarms and the influence of multiple trackers on load balance. The availability of content was measured in [51], which stated that a small fraction (23.5 percent) of peers could complete the download without the presence of seeds. By studying the performance of a number of BitTorrent swarms, Legout et al. [54] argued that the BitTorrent's built-in block scheduling policy and incentive mechanism are enough to guarantee a good performance. The resource supplied and consumed by users in several BitTorrent communities were examined in [11], and it was found in this study that users who supplied more resources tended to consume more. Stutzbach and Rejaie [94] studied the characteristics of churn of peers in several peer-to-peer systems, and found that most active peers were stable while other peers exhibited a high churn rate.

The recent trends on BitTorrent traffic and the topology over which BitTorrent traffic flows were comprehensively investigated in [77, 88]. Although both studies indicated that there was a moderate decrease in the fraction of Internet traffic accounted for by BitTorrent, it was also pointed out in [88] that BitTorrent still generated the most Internet traffic than any other Internet applications in all continents except South America, where BitTorrent is only second to another peer-to-peer file sharing application, Ares [1]. The torrent popularity and content distributed in BitTorrent swarms was intensively explored in [8] based on the information of 2.7m torrents hosted in the largest BitTorrent tracker, PublicBT tracker. It was revealed in [8] that a small number of torrents accounted for a significantly large proportion of BitTorrent users, and the most popular contents were movie, pornography and television, which corresponded to 11.2m, 3.2m and 2.4m seeders and leechers, respectively.

2.2.3 Protocol design and improvement

In what follows, we will review the studies on the protocol design and improvement of BitTorrent from four perspectives: block scheduling algorithm, incentives, topology awareness, content bundling and extensions to peer-to-peer streaming.

2.2.3.1 Block scheduling

Although BitTorrent was proven to be effective in terms of bandwidth utilization in many studies [83, 54, 14, 29, 81], there were still some studies aiming to improve block scheduling algorithm for a better download performance. Chan et al. [21] presented a graph-based dynamic weighted maximum-flow algorithm trying to distribute data as much as possible among peers in each cycle given the constraints of bandwidth and block distribution information of peers. Wu et al. [101] proposed a block scheduling policy which gives the highest priority to blocks desired by the neighbors with most blocks. A Proportional Fair Scheduling algorithm was presented in [74] which can be deployed in the seeders to accelerate the block distribution in BitTorrent swarms.

Some other studies aimed to decrease the average download time of peers in a BitTorrent swarm by strategically selecting the service receivers. A swarm partition algorithm was presented in [60] to decrease the average download time by grouping the peers in a BitTorrent swarm into several disjoint sets according to their bandwidth characteristics and disabling data exchange between peers in different sets. In [44], an adaptive neighbor selection strategy was presented which provides more opportunity for being unchoked[1] to the fresh peers than the tit-for-tat policy of BitTorrent.

2.2.3.2 Fairness and incentives

The issues regarding the fairness and incentives of BitTorrent have been intensively studied. Although it was argued in [27, 54] that the built-in tit-for-tat incentive mechanism of BitTorrent works fine, numerous studies indicated that BitTorrent cannot provide fairness to peers and is vulnerable to the strategic behavior of BitTorrent peers. Jun and Ahamad [48] stated that BitTorrent does not reward and punish peers properly, which results in free riding. In [61], three selfish behaviors, which are downloading only from seeds, downloading only from the fastest peers and advertising false blocks, were implemented, and their effectiveness was also evaluated. Although it appears that BitTorrent is resistant to these three exploits, a new strategic behavior was devised in [92, 67] which successfully renders it feasible to free ride in BitTorrent swarms. It was reported in these two studies that by connecting to much more neighbors than default value, a peer can finish the download without contributing to others. Although it was pointed out in [57, 67] that in swarms with leechers accounting for the most population, free-riders would perceive a much poorer download performance than the non-free-riders, Sirivianos et al. [92], however, demonstrated that even there is only one seed, a free-rider can perceive almost the same download performance as the non-free-riders by connecting to all the peers in the swarm.

[1] A peer unchoking a neighbor means that this peer is able to accept and serve the download request from this neighbor.

There also existed some studies exploiting the incentives mechanism of BitTorrent, i.e., tit-for-tat, to achieve a better download performance. A BitTorrent variation, BitTyant, was presented in [79] which enables peers to increase download rate by dynamically adjusting the number of neighbors to be unchoked and bandwidth allocated to each active connection. Two other mechanisms presented in [56] can also be leveraged to achieve a better download performance. They are: (1) intelligently underreporting the block availability information to prolong the interest, and (2) rewarding each unchoked neighbor with a proportional bandwidth share.

Many incentive mechanisms have been proposed to enhance the fairness and deter free riders. A family of incentive techniques, including discriminating server selection, maxflow-based subjective reputation and adaptive stranger policies, were proposed in [37], and it was argued that a combination of these techniques is able to foster the cooperation among rational peers. Two price-based mechanisms were presented in [33] to improve the fairness. With the proposed mechanisms in [33], peers would receive a download performance proportional to their upload capacity. A block-based tit-for-tat policy was presented in [13], which stipulates that a peer unchokes a neighbor only when the number of blocks that this peer contributed to this neighbor does not exceed the sum of the number of blocks that this peer downloaded from this neighbor and a specific parameter. The proportional share policy, i.e., a peer rewarding a neighbor with a bandwidth share proportional to the bandwidth share previously allocated to this peer by this neighbor, was also claimed to be able to achieve fairness and robustness [56]. A one hop reputation protocol, which propagates the reputation through at most one level of intermediary to extend the knowledge for peers to make decisions on whether to cooperate, was presented in [80]. In order to prevent free riders from downloading too much data from seeders[1], Chow et al. [26] presented a simple approach which forces the seeders to serve leechers with a specific fraction of blocks to realize a more

[1] A seeder serves the leechers which could download at the highest rate from it, or in a round robin manner, both of which are exploitable to free riders.

intelligent usage of seeders' capacity. In general, the enhancement in fairness implies a degradation in swarming efficient [13, 59], and the trade-off between the fairness and efficiency was extensively discussed in [59, 35].

2.2.3.3 Topology-awareness

Due to high throughput generated by BitTorrent, many studies have been conducted recently to enhance the BitTorrent locality and thus reduce inter-ISP traffic. Cuevas et al. [28] estimated to what degree the inter-ISP traffic can be reduced by studying millions of BitTorrent peers distributed in 11K autonomous systems, and argued that more than half of inter-ISP traffic can be saved at a cost of a less than 6% increase in the download time. In general, in order to decrease the inter-ISP traffic generated by BitTorrent peers, an intuitive way is to enable BitTorrent peers to establish neighborhood with peers in the same ISP. Different approaches have been proposed to achieve this goal. In [15, 16, 62], BitTorrent trackers are configured to return the peers in the same ISP as the requester in response to the announcement from the requester. In addition, Liu et al. also [62] presented some modifications to block scheduling and neighbor unchoking algorithms of BitTorrent, in which a peer unchokes the neighbors topologically closest to it and also downloads a piece from the closest unchoked neighbor with this piece. A topology-aware BitTorrent client, TopBT, was developed in [85], which leverages some network tools, e.g., ping, and tracert, to discover the network proximity among different peers and connect to topologically proximate peers. A novel CDN-based peer selection policy is presented in [24] which enables a peer to connect to those exhibiting similar CDN redirection behavior. All of these studies have shown that their approaches can not only reduce the inter-ISP, but also improve the download performance perceived by peers.

2.2.3.4 Content bundling

Content bundling is effective in terms of improving the availability of content in BitTorrent swarms by attracting more peers to join the swarm and prolonging the download time. Han et al. [41] argued that many contents can be bundled together due to the high similarity among them. It was further pointed out in [43] that content bundling is now very common in BitTorrent swarms, and the impact of content bundling on the performance metrics of BitTorrent swarms was empirically studied in [42]. [72] investigated the bundling strategies for publishers, and [55], on the other hand, carried out an analysis on the file selection strategies for peers in bundled swarms. In addition, a dynamic bundling system which requires peers to download the files in additional to those of their interest to enhance the content availability was presented in [103].

2.2.3.5 Extensions to peer-to-peer streaming

Due to the high bandwidth utilization, there were many studies on the feasibility of streaming media over BitTorrent. Several studies [19, 99, 90] have modified the rarest-first block scheduling algorithm to make it accountable for the playback deadline of data blocks and thus support the live streaming applications. Some other studies [68, 30, 89] were carried out to leverage BitTorrent to deliver on-demand media to multiple peers, and the corresponding design space was studied in [78] in detail.

2.3 The applications of network coding to peer-to-peer networks

Network coding was originally proposed in information theory [10], and thereafter it was introduced to peer-to-peer networks [25] [49]. Since then, network coding has gradually demonstrated its power in improving the overall performance of peer-to-peer networks. In [38], a file sharing protocol called Avalanche was proposed, which uses random linear network coding to accelerate the download process and facilitate the

block scheduling among neighbors. In Avalanche, the content publisher first generates at least n coded blocks, where n is the number of plain (original) blocks, and each coded block is a linear combination of all plain blocks, with the coding coefficients randomly selected from a finite field. The intermediate peers, i.e., those other than the publisher, are also able to generate new coded blocks. Upon receiving a coded block, an intermediate peer can generate a new coded block by computing a linear combination of all the coded blocks it has, with the coding coefficients also randomly selected from the finite field. In [100], Wang et al. proposed a live peer-to-peer streaming protocol, R^2, which also implemented random network coding to enable the coded blocks to be randomly push between neighbors without block availability information. There are also some commercial peer-to-peer streaming softwares [65] using network coding for effective content distribution.

The random network coding adopted in Avalanche [38] can also improve content availability. With the random linear network coding adopted in [38], each coded block is a linear combination of all plain blocks of the file, and the original file can be recovered as long as the dimension of the space spanned by the coding coefficient vectors of the coded blocks in the swarm is equal to the number of plain blocks, which occurs with high probability even the number of coding coefficient vectors is exactly the same as that of original blocks.

However, as we will show, with regard to the coding complexity, the coding strategy in Avalanche is not feasible for implementation. To alleviate the overhead, a sparse form of random linear network coding is proposed in [76], in which only two random plain blocks are used to generate new blocks. Nevertheless, due to the slow generation rate of new blocks and the limited diversity of blocks, the potential of improving content availability of this scheme cannot be fully exploited.

2.4 Private BitTorrent community

As a normal public BitTorrent community, a private BitTorrent community also provides two main services. It serves as an HTTP server hosting the torrent files for its users to download, and has a BitTorrent tracker in responsible for the organization and maintenance of the BitTorrent swarms. In contrast to the public BitTorrent communities which allow an arbitrary user to access, the private community[1] only serves the users who have already registered with the community. An outside user can join a private BitTorrent community by registering an account when the community opens up for registration, or getting invited by a registered user of the community. In general, it is not difficult to gain access to private communities, but for those high-level private communities, the account is very hard to obtain since the channel for free registration is closed and the registered users tend to send the invitations to those who are capable to justify the potential for making contribution to the community. Moreover, many private communities only allow the registered users with significant contributions to the community (e.g., those with high upload traffic) to give out invitations, which also increases the difficulty for outside users to join the community.

In BitTorrent swarms, peers periodically report the upload and download traffic to the tracker by requesting the tracker announce url, and in response to the request, the tracker would return a random list of online peers to the requesting peer. Thus it is necessary for the private BitTorrent tracker to reject the invalid requests and thereby not disclose the information of registered users. To this end, almost all private BitTorrent communities import a passkey system. Each valid user, i.e., registered user, is assigned with a unique passkey, and when a registered user downloads a torrent, its passkey is imprinted in the tracker announce url of the torrent. In this way, the private tracker is able to correctly identify the valid users by verifying the passkey in

[1]We may sometimes use "community" or "private community" to refer to "private BitTorrent community" for the ease of expression, if without ambiguity.

the announce url.

For each user[1], the private BitTorrent community keeps tracks of the accumulative upload and download traffic, as well as the accumulative seeding and downloading time, to determine the contribution of this user. Many private communities require users to maintain a minimum share ratio, namely the ratio of upload traffic to download traffic, to prevent from being warned or banned, which is known as SRE (share ratio enforcement). Due to the existence of SRE, many users choose to seed the leechers after completing the download in the hope of improving the share ratio, which leads to a high ratio of seeders to leechers and enhances the availability of the distributed files as well. SRE, however, may also render the survival of a part of users (e.g., the ADSL users with the upload bandwidth far less than the download bandwidth) difficult as the upload rates of these users are very slow. In order to improve this situation, many private communities offer some preferential torrents. When downloading these torrents, the download traffic would be counted at a discount, while the upload traffic would be counted normally.

In addition to SRE, there are another two common measures to encourage seeding in private BitTorrent communities. The first one is to reward seeding peers with points which can be used to trade for invitations and upload traffic, and the second measure is that by improving the upload traffic and share ratio, users can be entitled to multiple privileges, e.g., unlimited download slots, uploading torrents, giving out invitations, protecting the profile from being accessed by other users.

Due to its increasing prevalence, private BitTorrent communities received much attention recently. A couple of studies were carried out to characterize these communities. In the recent work [102, 64, 22, 73], it was found that in private BitTorrent communities, the ratio of seeder to leechers is much higher than that in public communities, and that compared with users in public communities, users in private BitTorrent communi-

[1]If no ambiguity raises, the term "user" will be used when we refer to the "register user" for the simplicity of expression.

ties tend to perceive a better download performance, and meanwhile maintain a higher share ratio. [64] and [22] also explored the effectiveness of share ratio enforcement (SRE), a policy commonly adopted in private BitTorrent communities, in improving the user share ratio. However, it was stated in [47] that the existence of SRE may result in users having to seed for an extremely long time in order to keep their share ratio upon a specific level. Different strategies were proposed in [46] for users of private BitTorrent communities to effectively improve their share ratios. An economic explanation to private BitTorrent communities was presented in [50]. In addition, targeting on the phenomenon that some malicious users may cheat the tracker by reporting a modified amount of traffic, Liu et al. [64] further presented an upload entropy scheme for deterring collusion among users. However, in these studies, only limited information about user traffic was presented, and the other two important respects of user behavior: seeding and leeching time, and download history, were not investigated. In addition, in the existing literature, how user behavior is influenced by user age and bandwidth was still unknown, as well as the impact of the content size and popularity of torrents on user behavior.

Chapter 3

Modeling and Analysis of Content Availability and Bundling in BitTorrent-like File Swarming Systems

3.1 Introduction

The content availability has been extensively studied in [70]. However, most of the results in [70] were derived based on the assumption that the population threshold, the minimum number of leechers to have all chunks, is equal to one, which is not realistic since a peer could complete the download only when there is at least one seed or two leechers. In this chapter, we extend [70] by re-deriving the performance metrics related to content availability under a more realistic assumption that the population threshold is larger than one. In particular, we first re-derive the closed-form expression for the duration of active periods using the result of residual busy period in [70], and then quantify the content availability and average sojourn time of peers based on the derived result of the duration of active periods. Moreover, we also derive the closed-form expression for the duration of the residual active period[1] in the swarms where peers continue to serve other peers after they complete the download, by modeling such swarms as an open Jackson network.

We also carry out an investigation on the impact of bundling on the performance of BitTorrent swarms. We quantify the impact of bundling on the length of the residual active period in the swarms with and without peers' altruistic behavior, and show that in the swarms without peers' altruistic behavior, the increase in the length of the

[1]The residual active period is the interval between the departure of publisher, i.e., the seed initiating the swarm, and the time when the content becomes unavailable, when the publisher will not return to the swarm. The residual active period has the same meaning of the residual busy period in [70].

residual active period resulted from bundling N files is strictly larger than N, and that in swarms with peers' altruistic behavior, bundling N files could increase the length of the residual active period by a factor of $e^{\Theta(N^2)}$. In addition, we also demonstrate that given an appropriate number of files are bundled in the swarm, the average sojourn time of peers can be reduced, as a result of the improvement in content availability. We finally perform extensive simulations to examine the validity of our theoretical analysis, and the simulation results exhibit a high conformity to our theoretical analysis.

The remainder of this chapter is organized as follows. We study the performance metrics related to content availability of BitTorrent swarms on the basis of some simple models in Section 3.2, and then carry out extensive simulations to validate our analysis in Section 3.3. Finally, we conclude this chapter in Section 3.4.

3.2 Models

3.2.1 Model description

In our model, the notations shown in Table 3.1 are introduced to describe the characteristics of BitTorrent swarms. In addition, to simplify our analysis, we make the following assumptions:

1. Peers arrive at the swarm according to a Poisson process.
2. The download time of each peer is exponentially distributed with the mean of $\frac{s}{u}$.
3. The download proceeds only in active periods, and is interrupted in passive periods.[1]. The interrupted download will resume when the publisher re-enters the swarm initiating another active period.
4. Peers depart immediately after the completion of download; this assumption will be relaxed in Section 3.2.2.4, in which peers' altruistic behaviors are taken into consideration.

[1] In fact, download can still proceed in passive periods as long as different peers have different chunks; however, it will sooner or later get stuck until the publisher enters the swarm, and the memoryless property guarantees that the residual download time has the same distribution of the download time. Therefore, assuming the download process is interrupted in passive periods has no impact on our analysis.

Table 3.1: Mathematical notations and their meanings

notation	meaning
λ	the arrival rate of new peers.
$\frac{1}{\gamma}$	the mean duration of publisher idle period, i.e., the interval between the publisher's departure and its next arrival, we assume the duration is an exponentially distributed random variable.
$\frac{1}{\beta}$	the mean of publisher residence time.
u	average download rate of peers.
s	size of the file to be distributed.
r	population threshold, the minimum number of online leechers which are enough to have all chunks.[1]
R	the length of the residual active period after the departure of the publisher, given that the publisher will not return, i.e. $\frac{1}{\gamma} \to \infty$; the bundled counterpart is denoted by $\mathcal{R}$.
A	the duration of an active period; the bundled counterpart is denoted by $\mathcal{A}$.
P	content unavailability; the bundled counterpart is denoted by $\mathcal{P}$.
T	average sojourn time of peers; the bundled counterpart is denoted by $\mathcal{T}$.

[1] According to BitTorrent specifications, peers always preferentially download the blocks which are least distributed among the neighbors. In this regard, the download behavior of peers, and hence the distribution of the blocks they have are determinate. Therefore, it is likely to determine the availability of content given the number of leechers.

5. Under the above assumptions, a BitTorrent swarm can be viewed as an M/M/∞ queue. We assume that this queue is in steady state when the publisher departs.[1]

6. A peer departs the swarm only after it completes the download, i.e., no peers abort the download.

7. The bundled files are of the same size, and bundling N files will cause both the arrival rate and the mean of download time increasing N times.

8. Bundling files exerts no impact on the population threshold; the rationale behind this assumption is that bundling files will lead to peers spending more time on downloading, and thus a peer in the bundled swarm tends to hold more chunks than a peer in the individual swarm does.

3.2.2 Content availability

3.2.2.1 A model for active periods

Since content availability is the fraction of swarm lifespan accounted for by active periods, we need to derive an expression of the active period duration first. An active period, as mentioned before, is a period when at least one seed is available or no less than r peers exist in the swarm to guarantee no chunks are lost, and such a period begins during which the publisher enters the swarm and ends with a peer exiting the swarm leaving behind less than r peers and no seeds.

As peers immediately exit the swarm after the completion of download, there is at most one seed, i.e., the publisher, in the swarm. Once the publisher departs, if the number of remaining peers is greater than or equal to r, the swarm remains in the active period, otherwise it falls into a passive period. In the former case, we need to figure out the length of residual active period. To this end, we first consider the situation that the publisher will not enter the swarm again[2], i.e., $\frac{1}{\gamma} \to \infty$. Under this situation, the active period is terminated when the number of peers falls below r, and thus during

[1] In real-world BitTorrent systems, since the intention of the publisher is to distribute its own content, it is natural to assume that the publisher departs only when there have been multiple peers completing the download, at which time the swarm has run for a long time, and the corresponding queue model is thus highly likely in steady state.

[2] Unless otherwise stated, when we talk about the length of the residual active period after the departure of the publisher, we assume that the publisher will never return to the swarm, i.e., $\frac{1}{\gamma} \to \infty$.

the residual active period the peer population is always above level $r-1$. Assuming that peer population is in steady state when the publisher departs, we have

Lemma 3.1 *When $\frac{1}{\gamma} \to \infty$, the expected duration of the residual active period after the departure of the publisher in the individual swarm is*

$$E[R] = \sum_{i=r}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} \sum_{j=r-1}^{i-1} \frac{j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} \tag{3.1}$$

and in the swarm with N files bundled, the expected duration of the residual active period after the departure of the publisher is

$$E[\mathcal{R}] = \sum_{i=r}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} \sum_{j=r-1}^{i-1} \frac{j!}{N\lambda(\frac{N^2\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{N^2\lambda s}{u})^k}{k!} \tag{3.2}$$

Proof. Please refer to Appendix A.2.1.

We then continue to discuss the residual active period taking into consideration that the publisher will go back to the swarm after leaving. If the publisher rejoins the swarm before peer population drops below r, another active period is initiated by the arrival of the publisher and has the same distribution with the original one since we have assumed that the publisher's residence time is an exponentially distributed variable and the peer population is in steady state when the publisher departs. By assuming the length of the residual active period after the departure of the publisher fluctuates slightly around its mean, we have

Theorem 3.2 *The expected length of an active period in the individual swarm is*

$$E[A] = (\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]} - \frac{1}{\gamma} \tag{3.3}$$

and the expected length of an active period in the swarm with N files bundled is

$$E[\mathcal{A}] = (\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[\mathcal{R}]} - \frac{1}{\gamma} \tag{3.4}$$

where $E[R]$ and $E[\mathcal{R}]$ are given in equation (3.1) and (3.2), respectively.

Proof. Please refer to Appendix A.2.2.

We can find that the only difference between equation (3.3) and (3.4) lies in the mean of R, which means that the impact of bundling on $E[R]$ determines to what extent bundling can increase the expected length of active periods.

If the publisher will not return to the swarm once it departs, i.e., $\frac{1}{\gamma} \to \infty$, then the expected length of an active period, $E[A]$, is equal to $\frac{1}{\beta} + E[R]$, as can be deduced from the definition of the active period. This can also be derived from equation (3.3), since

$$\begin{aligned}\lim_{\frac{1}{\gamma}\to\infty} E[A] &= \lim_{\gamma\to 0} \frac{1}{\beta} e^{\gamma E[R]} + \lim_{\gamma\to 0}(e^{\gamma E[R]} - 1)\frac{1}{\gamma} \\ &= \frac{1}{\beta} + \gamma E[R]\frac{1}{\gamma} = \frac{1}{\beta} + E[R]\end{aligned}$$

The similar result also holds for the bundled swarm.

We further examine the impact of bundling on the duration of active periods by considering the situation that the publisher will not reenter the swarm after departure. By comparing equation (3.1) with equation (3.2), we have

Theorem 3.3 *When $\frac{1}{\gamma} \to \infty$, bundling N files could lead to the expected length of the residual active period increasing at least N times, i.e.,*

$$E[\mathcal{R}] > NE[R] \tag{3.5}$$

where $E[R]$ and $E[\mathcal{R}]$ are given in equation (3.1) and (3.2), respectively.

Proof. Please refer to Appendix A.2.3.

Combined with equation (3.3) and (3.4), it can be observed that in a swarm where the publisher will come back to the swarm after leaving, bundling N files can lead to a remarkable increase in the length of an active period. The reason behind this rapid increase is that bundling files can not only improve the popularity, i.e., the arrival rate of peers, but also increase the download time[1]. Therefore, compared with the individual swarm, there are much more peers in the bundled swarm, and thus peer population is less likely to drop below level r, which implies a longer active period.

Since the length of publisher idle period has the same mean in both individual swarms and bundled swarms, the fraction of swarm lifespan accounted for by active periods in the bundled swarm is thus much larger than that in the individual swarm, and the availability of content is therefore significantly improved by bundling.

3.2.2.2 A model for content availability

We have mentioned that content availability can be derived by calculating the fraction of swarm lifespan accounted for by active periods. As the expected length of an active period is given, and the length of publisher idle period is exponentially distributed with the mean of $\frac{1}{\gamma}$, it is easy to see that

Theorem 3.4 *Content availability in the individual swarm, P, is*

$$P = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]}} \tag{3.6}$$

and the availability of content in the swarm with N files bundled, $\mathcal{P}$, is

$$\mathcal{P} = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[\mathcal{R}]}} \tag{3.7}$$

where $E[R]$ and $E[\mathcal{R}]$ are given in equation (3.1) and (3.2), respectively.

[1]Note that the download time represents the time only spent in active periods. Thus the download time always increases as more files are bundled in the swarm; however, this may not hold for the sojourn time as we will show that, in swarms with content highly unavailable, bundling could reduce the average sojourn time, although the download time increases.

Proof. Please refer to Appendix A.2.4.

To appreciate to what extent bundling could improve content availability, we consider two swarms with different levels of publisher availability. The publisher in the first swarm is highly available, i.e., $\frac{1}{\beta} \gg \frac{1}{\gamma}$, while the publisher in the second swarm is highly unavailable, i.e., $\frac{1}{\gamma} \gg E[R]$, which also implies $\frac{1}{\gamma} \gg \frac{1}{\beta}$. For content availability in the first swarm, we have

$$P = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]}} \approx 1 - \frac{1}{\frac{\gamma}{\beta}e^{\gamma E[R]}} > 1 - \frac{\beta}{\gamma}$$

As $\frac{1}{\beta} \gg \frac{1}{\gamma}$, the content availability, P, nearly equals to 1. Therefore, there is little space for improvement of content availability in a swarm with high publisher availability.

In the second swarm where the publisher is highly unavailable, the content availability is given by

$$P = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]}} \approx \frac{e^{\gamma E[R]} - 1}{e^{\gamma E[R]}}$$

and if N files are bundled in this swarm, the content availability is then given by

$$\mathcal{P} = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[\mathcal{R}]}} \approx \frac{e^{\gamma E[\mathcal{R}]} - 1}{e^{\gamma E[\mathcal{R}]}}$$

Thus, it follows that

$$\frac{\mathcal{P}}{P} \approx \frac{(e^{\gamma E[\mathcal{R}]} - 1)/e^{\gamma E[\mathcal{R}]}}{(e^{\gamma E[R]} - 1)/e^{\gamma E[R]}} = \frac{e^{\gamma E[R]}(e^{\gamma E[\mathcal{R}]} - 1)}{e^{\gamma E[\mathcal{R}]}(e^{\gamma E[R]} - 1)} \approx \frac{e^{\gamma E[\mathcal{R}]} - 1}{\gamma E[R] e^{\gamma E[\mathcal{R}]}}$$

As $E[\mathcal{R}]$ increases at least N times by bundling N files, when the number of bundled files, N, is large enough, we then have $\mathcal{P} \approx \frac{e^{\gamma E[\mathcal{R}]}-1}{e^{\gamma E[\mathcal{R}]}} \approx 1$, and thus $\frac{\mathcal{P}}{P} \approx \frac{1}{\gamma E[R]}$. Therefore, the improvement of content availability is very remarkable when $\frac{1}{\gamma} \gg E[R]$, and it also shows that, as long as enough files are bundled, the content availability will tend to 1, regardless of publisher availability.

3.2.2.3 A model for peer sojourn time

Once a swarm is initiated by the publisher, interested peers then join the swarm to download the file, and depart the swarm after completing their downloads. As the publisher owns all the content, a swarm begins with an active period, and enters a passive period when a peer departs the swarm and leaves behind no seeds and not enough peers to have all the chunks. When the publisher enters the swarm again[1], the passive period is terminated and another active period is initiated. Therefore, the lifetime of a swarm always alternates between active periods and passive periods, and the sojourn time of a peer in the swarm consists of two parts: the time spent in active periods and the time spent in passive periods.

Peers may come to the swarm in either an active period or a passive period. For peers arriving in a passive period, they must wait for the publisher to become available, and for those arriving in an active period, they will also spend some time in passive periods if they cannot complete their downloads in the active period. According to the definition of the active period, at most $r-1$ peers are left when the swarm enters a passive period. Therefore, the fraction of peers accounted for by those who cannot finish the download in an active period is about $\frac{r-1}{C}$, where C represents the number of peers appearing in the same active period.

As the *rarest-first* policy is used in BitTorrent to determine which chunk to be requested, chunks tend to be uniformly distributed among peers, and a small number of peers can own all the chunks, which means r takes a small value. Therefore, we can simply assume that $r \ll C$, and thus neglect the impact of peers crossing two or more active periods on the analysis of average sojourn time. In other words, the sojourn time of a peer mainly depends on the download time and the waiting time before starting download. The former is exponentially distributed with the mean of $\frac{s}{u}$, and the latter

[1] We omit the possibility that other peers will re-enter the swarm after completing their downloads as this is unlikely to happen.

is also exponentially distributed with the mean of $\frac{1}{\gamma}$ if this peer arrives in a passive period and is 0 if this peer arrives in an active period.

The mean sojourn time is thus given by:

Theorem 3.5 *The mean of peer sojourn time in the individual swarm, $E[T]$, is*

$$E[T] = \frac{s}{u} + \frac{\beta/\gamma}{(\beta+\gamma)e^{\gamma E[R]}} \tag{3.8}$$

and the mean of peer sojourn time in the swarm with N files bundled, $E[\mathcal{T}]$, is

$$E[\mathcal{T}] = \frac{Ns}{u} + \frac{\beta/\gamma}{(\beta+\gamma)e^{\gamma E[\mathcal{R}]}} \tag{3.9}$$

where $E[R]$ and $E[\mathcal{R}]$ are given in equation (3.1) and (3.2), respectively.

Proof. Please refer to Appendix A.2.5.

We also investigate under what condition peers can benefit from bundling with regard to average sojourn time.

Theorem 3.6 *The average sojourn time of peers can be reduced if the number of bundled files satisfies the following condition:*

$$2 \leq N \leq \frac{\beta/\gamma}{(\beta+\gamma)e^{\gamma E[R]}} \left(1 - e^{-\gamma E[R]}\right) \frac{u}{s} + 1 \tag{3.10}$$

where $E[R]$ is given in equation (3.1).

Proof. Please refer to Appendix A.2.6.

3.2.2.4 Altruistic Behavior

A peer may continue to stay in the swarm after completion of the download, and upload to other peers as a seed. We assume the length of this seeding period is an

exponentially distributed variable with a mean of $\frac{1}{\theta}$. In this case, the swarm can be viewed as a Jackson network consisting of two M/M/∞ queues. The first queue consists of leechers, i.e., peers who have not finished the download, and the second queue consists of all the seeds other than the publisher; the output of the first queue is the input of the second queue. According to Burke's Theorem [17], the output of an M/M/∞ queue in steady state is a Poisson process, and the departure rate is same as the arrival rate. We also assume the Jackson network is in steady state when the publisher departs.

Lemma 3.7 *When $\frac{1}{\gamma} \to \infty$, in the individual swarm, the expected length of residual active period after the departure of the publisher is*

$$E[R] = \sum_{i=1}^{\infty} \frac{e^{-\lambda/\theta}(\frac{\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \frac{j!}{\lambda(\frac{\lambda}{\theta})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda}{\theta})^k}{k!} + \frac{1-P_r}{P_r} \frac{e^{\lambda/\theta}}{\lambda} \tag{3.11}$$

where $P_r = e^{-\lambda s/u} \sum_{i=0}^{r-1} \frac{(\lambda s/u)^i}{i!}$, and in the swarm with N files bundled, we have

$$E[\mathcal{R}] = \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \frac{j!}{N\lambda(\frac{N\lambda}{\theta})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{N\lambda}{\theta})^k}{k!} + \frac{1-\mathcal{P}_r}{\mathcal{P}_r} \frac{e^{N\lambda/\theta}}{N\lambda} \tag{3.12}$$

where $\mathcal{P}_r = e^{-N^2\lambda s/u} \sum_{i=0}^{r-1} \frac{(N^2\lambda s/u)^i}{i!}$.

Proof. Please refer to Appendix A.2.7.

We can derive the expression of other metrics, e.g., content availability and average peer sojourn time, in the swarms with altruistic peers by using the above two equations.

We also want to examine the impact of bundling on $E[R]$ in swarms with altruistic peers, and this can be accomplished by comparing equation (3.11) with (3.12).

Theorem 3.8 *When $\frac{1}{\gamma} \to \infty$, bundling N files could lead to an increase of $e^{\Theta(N^2)}$ on $E[R]$ in the presence of peers' altruistic behavior, i.e.,*

$$\log \frac{E[\mathcal{R}]}{E[R]} = \Theta(N^2) \tag{3.13}$$

where $E[R]$ and $E[\mathcal{R}]$ are given in equation (3.11) and (3.12), respectively.

Proof. Please refer to Appendix A.2.8.

3.3 Simulation

In order to verify our analytical results, we have performed extensive simulations using the BitTorrent simulator [32], which is built on Network Simulator [6], and the simulation results exhibit a high conformity to our theoretical analysis.

3.3.1 Experimental Setup

In [70], a file with a size of only 4MB is disseminated in the individual swarm, and this could lead to the download process suffering a high uncertainty as the download time may highly depend on the network condition, e.g., network congestion, rather than peer arrival rate and access bandwidth. In order to decrease the influence of the network condition on peers' download time, in the simulation, we deploy a BitTorrent swarm where a file with a size of 100MB is distributed, and 30,000 peers with a homogeneous access bandwidth of 50KB/s arrive according to a Poisson process. The publisher's residence time is exponentially distributed with a mean of 5 hours, i.e., 18,000s, which is long enough for the swarm entering the steady state. We run the simulation for many times, and in each running, the arrival rate of peers and the value of $\frac{1}{\gamma}$, i.e, the mean length of publisher idle period, are adjusted to explore the performance variance of the swarm under different conditions.

As there are many control messages, other than chunks, to be transmitted in BitTorrent swarms, and the upload bandwidth of newly participating peers cannot be effectively utilized, the practical download time of peers is slightly longer than the theoretical result, s/u, which takes the value of $100000/50 = 2000s$ in the simulations. According to the simulation results, the residence time of peers is about 2500 seconds given that the content is available. Therefore, in the calculation of proposed

expressions, we also set the value of s/u to 2500 seconds in order to keep consistent with simulation.[1][2] It remains to determine the value of population threshold, r. The simulations indicate that $\frac{\lambda \times s}{u}$ is a good estimation of r, where s, u, λ denote the file size, access bandwidth and peer arrival rate, respectively.[3] Since in our settings, $\frac{\lambda \times s}{u}$ fluctuates around 30 in all swarms, we let the population threshold take the value of 30 for simplicity.

3.3.2 Active Periods

The analysis of active periods plays a central role in our work, as the derivations of other performance metrics are all based on this analysis. Thus the accuracy of the analysis of active periods to a large extent determines the validity of our overall analysis.

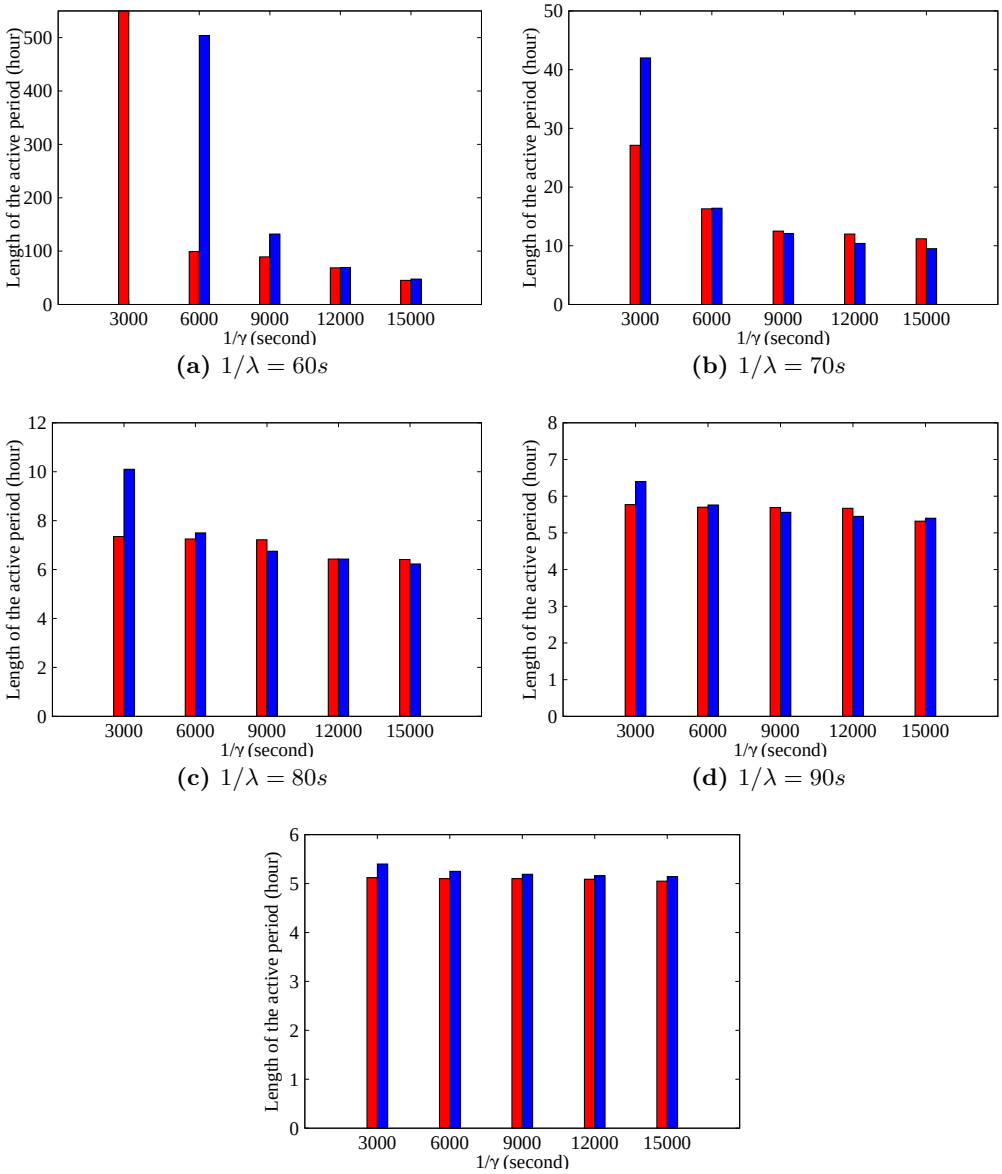
Fig. 3.1 demonstrates both the simulation result and the theoretical result of the length of active periods under different combinations of peer inter-arrival interval, $1/\lambda$, and the length of publish idle period, $1/\gamma$. It can be clearly observed from this figure that under most scenarios, the length of active periods obtained from simulations is roughly the same as that obtained from equation (3.3), which indicates that the result obtained from our theoretical analysis is a good approximation to the simulation result. Another observation can be drawn from this figure is that both the theoretical result and the simulation result of the length of active periods vary in the same way as $1/\lambda$ and $1/\gamma$ change.

However, there also exist some scenarios in which the simulation result exhibits a manifest difference from the theoretical result. These scenarios are: (1) $1/\lambda = 60s, 1/\gamma = 3000s$, (2) $1/\lambda = 60s, 1/\gamma = 6000s$, (3) $1/\lambda = 70s, 1/\gamma = 3000s$ and (4)

[1] Although the adjustment of s/u may vary in different system settings, the adjustment we did for our simulation is a good indication.

[2] The assumption that the download time of peers is exponentially distributed is relaxed in the simulation.

[3] This formula for population threshold may not be applicable to other system settings, as the population threshold can be affected by these factors, i.e., file size, access bandwidth and arrival rate, in other manners. Nevertheless, we believe this formula can provide some hints for the estimation of the population threshold.



(a) $1/\lambda = 60s$ **(b)** $1/\lambda = 70s$

(c) $1/\lambda = 80s$ **(d)** $1/\lambda = 90s$

(e) $1/\lambda = 100s$

Figure 3.1: The length of active periods under different scenarios (The red and blue columns represent the simulation result and theoretical result, respectively)

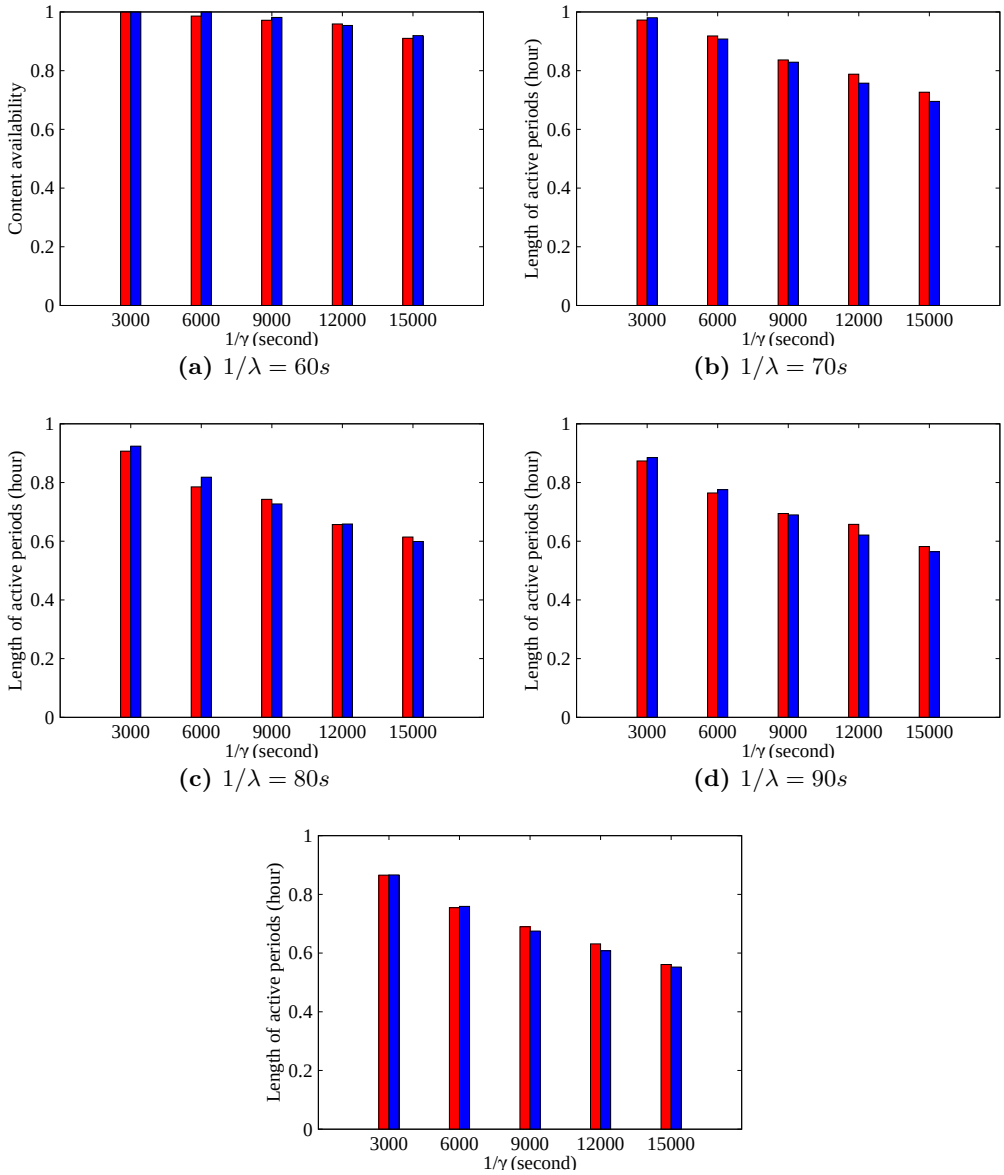
$1/\lambda = 80s, 1/\gamma = 3000s$.

For the first case, the theoretical result is 33608, and is not shown in Fig. 3.1.a since it is too large. In fact, the real difference between the simulation result and theoretical result in this case is much smaller than that shown in Fig. 3.1. The total running time of the simulation corresponding with the first case is 500 hours, which is the product of peer inter-arrival interval, $60s$ and peer population, 30000. As shown in Fig. 3.1.a, the simulation result of the length of active periods is also 500 hours, which implies that there is only one active period through the simulation. We also find from this simulation that only one peer cannot complete the download since all other peers immediately departs after completing the download. Thus we can say that the only active period is initiated by the publisher and terminated at the epoch when the last departing peer left behind only one peer. In other words, the swarm will remain in the active period as long as peers continue to arrive at the same rate.

It can be shown that the difference between the simulation result and the theoretical result only appears in scenarios with shorter publisher idle period. This may be because $E[A]$ is an exponential function of the length of publisher idle period, and thus a slightly shorter publisher idle period could lead to a dramatic increase of $E[A]$.

3.3.3 Content Availability

The content availability of swarms under different scenarios is shown in Fig. 3.2. It is straightforward to observe from the figure that the result measured from simulation is almost the same as that derived from the theoretical analysis, which justifies our analysis on the content availability. We can also observe that increasing the value of peer inter-arrival interval, $1/\lambda$, by 10 seconds has the similar effect on the content availability as increasing the value of the length of publisher idle period, $1/\gamma$, by 3000 seconds. This is because they play different roles in the equation (3.3) and (3.6), and thus exert different impact on the content availability.



(a) $1/\lambda = 60s$ **(b)** $1/\lambda = 70s$

(c) $1/\lambda = 80s$ **(d)** $1/\lambda = 90s$

(e) $1/\lambda = 100s$

Figure 3.2: Content availability under different scenarios (The red and blue columns represent the simulation result and theoretical result, respectively)

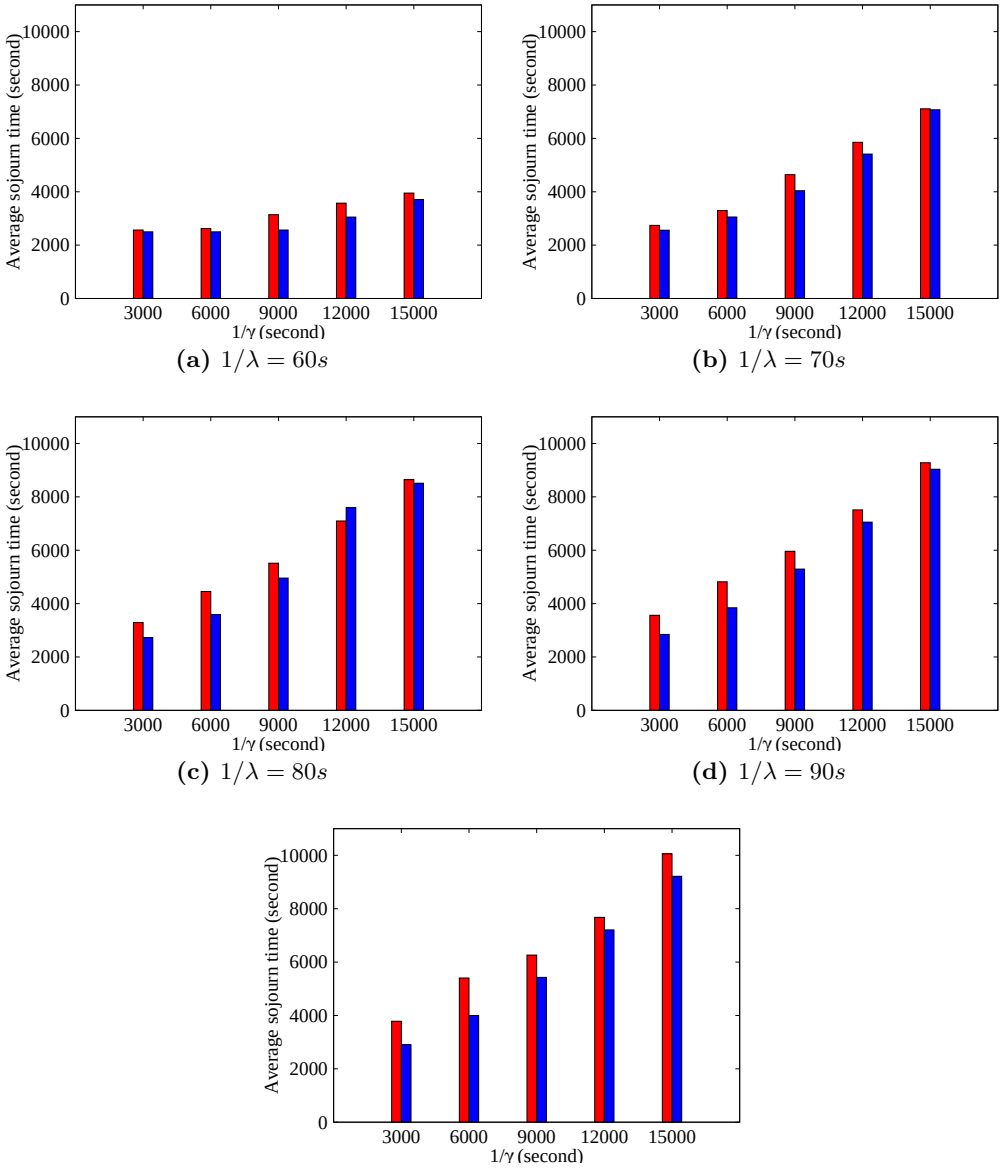
3.3.4 Average Sojourn Time

Fig. 3.3 illustrates the average sojourn time of peers under different scenarios. As shown in the figure, the theoretical results is a good estimation of the corresponding simulation result. We can also draw from the figure that in most scenarios, the theoretical result is slightly less than the corresponding simulation result. This can be explained in the following two aspects. First, as we mentioned above, there are many control messages transmitted in BitTorrent swarms, and the upload bandwidth of new participants cannot be effectively utilized as they own little chunks, both of which can extend the download process. Second, the number of passive periods is limited in the simulations in cases in which peer inter-arrival interval and the length of publisher idle period are small, and thus the average length of passive periods, which are exponentially distributed with the mean of $1/\gamma$, may exhibit a significant difference with its mean. The second explanation can also be inferred from the figure as we can see that as the increase of the length of public idle period, the difference between the simulation result and the corresponding theoretical result experiences a slight decrease.

3.3.5 The impact of bundling

We also have run simulations to validate our analysis of the impact of bundling on the performance metrics. In order to demonstrate the impact, we deploy a swarm with a high unavailability of content: The peer inter-arrival interval and the mean of publisher idle period, i.e., $1/\gamma$, are 600 seconds and 72000 seconds respectively, and the assumption that bundling N files increases peer arrival rate by a factor of N also applies in this section.

Fig. 3.4 shows the impact of bundling on the performance metrics of BitTorrent swarms. From this figure, we can see that when three files are bundled, the length of active periods is nearly 40 times larger than that in the individual swarm, and the content availability is also greatly improved to about 0.95. Furthermore, when three

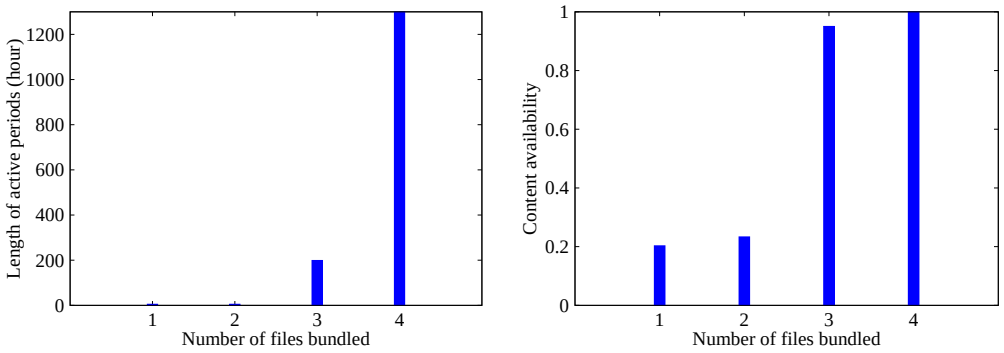


(a) $1/\lambda = 60s$ **(b)** $1/\lambda = 70s$

(c) $1/\lambda = 80s$ **(d)** $1/\lambda = 90s$

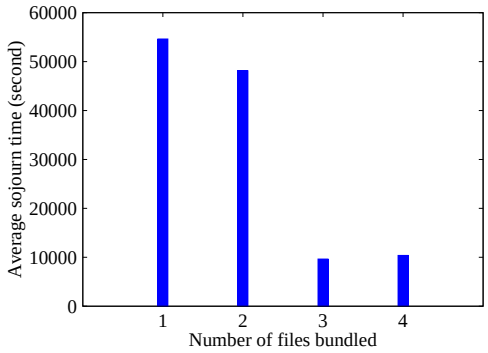
(e) $1/\lambda = 100s$

Figure 3.3: Average sojourn time of peers under different scenarios (The red and blue columns represent the simulation result and theoretical result, respectively)



(a) Length of the active period

(b) Content availability



(c) Average sojourn time

Figure 3.4: The impact of bundling on performance metrics of BitTorrent swarms

files are bundled, the average sojourn time of peers decreases significantly as a result of the enhanced content availability, although peers have to download the file with a size of 300MB. When four files are bundled, the content is always available throughout the simulation, [1] but the average sojourn time is increased contrast with that in the swarm with three files bundled, since in the swarm with three files bundled, as a result of high content availability, peers spend the majority of the time in downloading. All of these observations highly coincides with our analysis on the impact of bundling, which provides a positive evidence for the correctness of our discussion on the impact of bundling.

3.4 Conclusion

In this chapter, we have presented a re-derivation for the length of the active period, content availability and average peer sojourn time of BitTorrent swarms based on the result of the congestion period of the M/M/∞ queue [86] and the result of the residual busy period in [70]. We have also quantified the effect of bundling by showing that bundling N files could increase the length of the residual active period after the departure of the publisher by at least N times, and that the average peer sojourn time can be reduced by bundling an appropriate number of files. In addition, we also derive the length of the residual active period after the departure of the publisher and the improvement in this metric caused by bundling in the presence of peers' altruistic behavior. We finally validate our analysis through extensive simulations.

[1] There are 30,000 peers arriving at a rate of 1/150 peers/sec.

Chapter 4

Using Network Coding to Ameliorate the Content Availability of BitTorrent Swarms

4.1 Introduction

It has already been shown that network coding [10] can be used in peer-to-peer networks to accelerate the download process and facilitate block scheduling [38] [100]. Nevertheless, the impact of network coding on the availability of content is not well understood. Specifically, neither the probability of content being available when using network coding nor to what extent network coding improves content availability has been investigated. In this chapter, both metrics will be explored to reveal the influence that network coding exerts on content availability.

In this chapter, we aim to investigate the feasibility of using network coding to ameliorate the content availability of BitTorrent swarms. Two network coding schemes are discussed in this chapter. The first one is using all the original blocks to generate new blocks, which is adopted in [38], and the second one is a sparse form of random linear network coding proposed in [76], in which only two random plain blocks[1] are used to generate new blocks.

We first conduct theoretical analysis of the impact of the two network coding schemes on the content availability and bandwidth utilization as these two metrics exert an important influence on the overall performance of BitTorrent swarms. It is found that both schemes are able to not only increase the probability of content being available, but also improve the usability of peers, which implies a higher bandwidth utilization. Our analysis, however, also demonstrates that both schemes have their own

[1] Throughout the rest of chapter, we will use "original blocks" and "plain blocks" interchangeably.

drawbacks which degrade the feasibility of being incorporated into practical BitTorrent system. The first scheme, in which a coded block is a linear combination of all plain blocks, is infeasible in terms of the overhead incurred by the computation and disk operation. In the second coding scheme, the potential of improving content availability cannot be effectively exploited due to the slow generation rate of new blocks and the limited diversity of blocks.

In order to achieve a high content availability while keeping the incurred overhead at a low level, we propose another simple but very effective sparse network coding scheme which also uses only two plain blocks to generate new blocks. Different from the second coding scheme in which the new blocks are generated from two random plain blocks, our scheme stipulates that a plain block can only be combined with another fixed plain block to generate new blocks, and once a peer has downloaded two coded blocks with the same two underlying plain blocks, it can then reconstruct the two original blocks which can further be used to generate new coded blocks. In this way, the generation rate of new coded blocks is very fast, and the diversity of blocks is also enhanced.

As the transmission unit is a combination of plain blocks when network coding is imported, the original block scheduling policy in BitTorrent, *rarest-first*, no longer works. In order to keep the high bandwidth utilization, we propose a simple variant of the rarest-first policy in BitTorrent, which computes the importance of coded blocks according to their underlying plain blocks: those coded blocks consisting of plain blocks less distributed among neighbors will be assigned with a higher priority to be requested. We further differentiate among the coded blocks with the same underlying plain blocks but linearly independent coding vectors.

Through extensive simulations, we demonstrate that, compared with pure BitTorrent swarms, the number of peers needed to sustain the download process is much less when the proposed coding scheme is imported, which means our way of using network coding is very effective in terms of improving content availability of BitTorrent swarms.

Although the coding scheme may lead to a slight increase in the size of control messages, the average download time is hardly affected as the control messages account for only a small part of whole throughput, which is also confirmed in simulations. Furthermore, since each coded block is generated from two blocks, the number of disk operation is only twice the number in BitTorrent, and the complexity of computation involved in coding and decoding process for each peer is also far less than that in Avalanche[38], which render our coding scheme more realistic and feasible in applications. Compared with the coding scheme proposed in [76], our coding scheme, combined with the proposed block scheduling algorithm, enables new blocks to be generated at a higher rate, which in turn enhances the block diversity significantly.

The remainder of the chapter is organized as follows. In Section 4.2, we analyze the impact of network coding on the content availability of BitTorrent swarms and bandwidth utilization. A simple sparse network coding scheme is presented in Section 4.3. To make BitTorrent be able to effectively work with the proposed network coding scheme, in Section 4.4, we propose an algorithm for block scheduling. The improvement of content availability induced by the proposed coding scheme and algorithms is validated in Section 4.5. Finally, we conclude this chapter in Section 4.6.

4.2 Analyzing the Effect of Network Coding

Network coding is able to improve bandwidth utilization and content availability. While the improvement in bandwidth utilization is demonstrated in [38] and [100] through simulations and real traces, the fact that network coding can ameliorate the availability of content is not well understood. In this section, we present a theoretical analysis of the impact of network coding on the usability of coded blocks, which to a large degree determines the utilization of network resources, and the impact of network coding on the content availability is also explored.

4.2.1 Background

We first give a brief introduction to the two network coding schemes [38, 76] which will be discussed in our analysis. Consider a peer A wants to disseminate a file with n plain blocks. Denote the n plain blocks by a vector $b = [b_1, b_2, \ldots, b_n]$. If network coding is used, n coded blocks, from which the original file can be recovered, will be generated by peer A for distribution. Under both coding schemes, each coded block, $\boldsymbol{b}$, can be represented by a linear combination of the plain blocks, i.e.,

$$\boldsymbol{b} = \sum_{i=1}^{n} c_i b_i$$

where c_i, which is selected from a finite field, is the coding coefficient of the i-th plain block, b_i, and the vector $c = [c_i, c_2, \ldots, c_n]$ is the coding coefficient vector of coded block $\boldsymbol{b}$. The difference between the first and second coding scheme in this coding process lies in the number of non-zero coding coefficients. The first scheme [38] requires that there is at least one non-zero coding coefficient, while the second one [76] stipulates that there are exactly two non-zero coding coefficients. In order to recover the original file, the dimension of the space spanned by the set of coding coefficient vectors of n coded blocks must be equal to n.

Once a peer other than A has downloaded more than one coded blocks, it may be able to use these coded blocks to generate new coded blocks. For simplicity, we use the term *recoding process* to represent the generation of new coded blocks, i.e., the coded blocks generated by peers other than A. The recoding processes of the two coding schemes are also different. Under the first scheme, any two or more coded blocks can be used to generate coded blocks in the following way:

$$\boldsymbol{b'} = \sum_{i=1}^{k} c_i \boldsymbol{b}_i, 1 < k \leq n$$

where $\boldsymbol{b}_i$ is a coded block owned by this peer, with the coding coefficient being c_i, which is also randomly selected from the underlying finite field. Again, there is at least one non-zero coding coefficient in this equation. Under the second coding scheme, however, in order to guarantee that the newly generated coded blocks also contain the information of exactly two plain blocks, two coded blocks can be used to generate new coded blocks only if they share at least one common underlying plain block. More specifically, two coded blocks sharing one common plain block can be used to generate only one new coded block, and multiple new coded blocks can be obtained from different linear combinations of two coded blocks with the same two underlying plain blocks, provided that the coding coefficient vectors of the two coded blocks are linearly independent.

4.2.2 Analysis

Table 4.1: Mathematical notations and their meanings

notation	meaning
n	the number of plain blocks into which the original file is split, we assume n is even.
l	the number of peers in the swarm.
p_i	the number of blocks hosted at peer i.
$\mathbb{A}_{i,j}$	the event that the first peer who has downloaded i blocks is of no use to the second peer which has j blocks.

The mathematical notations that will be used in the following analysis are shown in Table 4.1. To study the effect of network coding, we further assume that the probability distribution of p_i, i.e., the number of blocks at peer i, is given by

$$\Pr(p_i = k) = \frac{1}{n+1}, \quad 1 \leq i \leq l, 0 \leq k \leq n \tag{4.1}$$

and the set of coding coefficient vectors of blocks at each peer is linearly independent.[1]

[1]For a plain block b_i, the i-th element of the corresponding coding coefficient vector is 1, and other elements are all zero.

Since all peers have the same probability distribution of the number of blocks, we will omit the subscript of p_i in circumstances where there is no ambiguity.

4.2.2.1 The Impact on the Usability of Peers

Suppose a peer has already downloaded i blocks. Without network coding, a plain block is usable to this peer with probability $\frac{n-i}{n}$, and a peer is of use to another peer with probability $\frac{\log n}{n}$ [83]. When network coding is used, a coded block is usable to this peer if this block is not a linear combination of the coded blocks hosted at this peer, which, as we will show later, occurs with a probability larger than $\frac{n-i}{n}$, even only two plain blocks are used to generate coded blocks.

We first consider the coding scheme in [38], in which a coded block contains the information of all plain blocks. In this case, we have

Lemma 4.1 *a peer with i coded blocks is unusable to a peer with j blocks, i.e., $\mathbb{A}_{i,j}$ occurs, with the probability:*

$$\Pr(\mathbb{A}_{i,j}) = \begin{cases} 1, & i = 0 \\ \prod_{k=0}^{i} \frac{q^j - q^k}{q^n - q^k}, & 0 < i \leq j \\ 0, & j < i \leq n \end{cases} \tag{4.2}$$

where q is the size of the underlying Galois field, and throughout the section, we assume $q > 2$.

Proof. Please refer to Appendix B.1.

Given equation (4.1) and (4.2), we can then derive the probability that a peer is of use to another peer.

Theorem 4.2 *The probability that a peer is usable to another one is larger than $\frac{n-1}{n+1}$.*

Proof. Please refer to Appendix B.2.

We then consider the case in which only two original blocks are used to generate new blocks. Under this coding scheme, the probability of $\mathbb{A}_{i,j}$ is given in the following lemma.

Lemma 4.3 *The probability of the event $\mathbb{A}_{i,j}$ occurring is*

$$\Pr(\mathbb{A}_{i,j}) \begin{cases} = 1, & i = 0 \\ \leq \left(\frac{j}{n}\right)^{2i}, & 0 < i \leq j \\ = 0, & j < i \leq n \end{cases} \tag{4.3}$$

Proof. Please refer to Appendix B.3.

Now we can derive the probability that a peer is usable to another peer when only two original blocks are combined to generate new blocks.

Theorem 4.4 *Under the coding scheme which generates new coded blocks using only two plain blocks, the probability that a peer is usable to another one is larger than $\frac{n-2}{n+1}$.*

Proof. Please refer to Appendix B.4.

From Theorem (4.2) and (4.4), it is straightforward to obtain that peer usability is very high under both coding schemes, which implies a high bandwidth utilization.

4.2.2.2 The Impact on the Availability of Content

We then conduct analysis on the probability of content being available using different network coding schemes. Since the content is trivially available if a peer has n blocks, we focus our attention on the situation that each peer has at most $n-1$ blocks, and the number of blocks at each peer is uniformly distributed in $[0, n-1]$. Suppose there are l peers in the swarm, and the distribution of blocks among peers is represented by a l-dimension vector $p = [p_1, p_2, \ldots, p_l]$, where p_i is the number of blocks at peer i. We further let P_p denote the probability of content being available given the block

distribution vector p. In order to distinguish among different coding schemes, we will refer to this probability as $\bar{P}_p$ when the first coding scheme is used, and when the second scheme is used, $\hat{P}_p$ will be used.

The probability of content being available when network coding is not used, P_p, can be obtained by simply using the inclusion-exclusion principle, which is shown as follows:

$$P_p = 1 - \sum_{i=1}^{n-1} (-1)^{i+1} \binom{n}{i} \prod_{j=1}^{l} \frac{\binom{n-i}{p_j}}{\binom{n}{p_j}}$$

with the convention that $\binom{i}{j} = 0$ when $i < j$.

We then take the two network coding schemes into consideration. For the first coding scheme, we first consider the case that $p_i = 1, 1 \leq i \leq l$, and in this case, we have

Lemma 4.5 *The probability of content being available, denoted by $\bar{P}_p$, is given as follows:*

$$\bar{P}_p \geq \max\left\{1 - \left(\frac{1}{q-1} - \frac{n}{q^n - 1}\right)^{\sum_{i=1}^{l} p_i - n + 1}, 0\right\} \tag{4.4}$$

Proof. Please refer to Appendix B.5.

We then extend the above result by allowing a peer to own more than one block. Given a block distribution vector $p = [p_1, p_2, \ldots, p_l]$, where $1 \leq p_i < n$, we can construct a vector $p' = [p'_1, p'_2, \ldots, p'_{\sum_{i=1}^{l} p_i}]$, which satisfies that $p'_i = 1$, and

Theorem 4.6

$$\bar{P}_p \geq \bar{P}_{p'} \geq \max\left\{1 - \left(\frac{1}{q-1} - \frac{n}{q^n - 1}\right)^{\sum_{i=1}^{l} p_i - n + 1}, 0\right\} \tag{4.5}$$

Proof. Please refer to Appendix B.6.

In addition, we also want to quantify to what degree the availability of content is enhanced by this network coding scheme.

Theorem 4.7 *For any block distribution vector p which satisfies $\sum_{i=1}^{l} p_i = m \geq n$, we have*

$$\frac{1-P_p}{1-\bar{P}_p} \geq (n-m+c(n-1))n^{-c}(q-1)^{m-n+1} \tag{4.6}$$

where $c = \lceil \frac{m}{n-1} \rceil - 1$.

Proof. Please refer to Appendix B.7.

Remark. From the above theorem we can easily derive that

$$\frac{1-P_p}{1-\bar{P}_p} \geq \frac{n-1}{n}(q-1)$$

since

$$\begin{aligned}
&\frac{1-P_p}{1-\bar{P}_p} \\
&\geq (n-m+c(n-1))n^{-c}(q-1)^{m-n+1} \\
&\geq (n-(c(n-1)+1)+c(n-1))n^{-c}(q-1)^{c(n-1)+1-(n-1)} \\
&= (n-1)(q-1)^{(c-1)(n-1)+1}n^{-c} \\
&\geq \frac{n-1}{n}(q-1)
\end{aligned}$$

We now consider the case in which the second coding scheme is used. Denote the probability of content being available given the block distribution vector p by $\hat{P}_p$. At this moment, we are only able to give the comparison between P_p and $\hat{P}_p$ when all the elements of p are equal to 1, and in this case, we have

Theorem 4.8

$$\hat{P}_p \geq P_p \tag{4.7}$$

Proof. Please refer to Appendix B.8.

Theorem (4.7) and (4.8) demonstrate that both coding schemes could improve the content availability, and the improvement induced by the first coding scheme is rather

substantial: if there are no less than n blocks in the swarm, using the first coding scheme could reduce the probability of content being unavailable by a factor of at least $\frac{(n-1)(q-1)}{n}$.

4.3 A Simple Sparse Network Coding Scheme

Although both network coding schemes exhibit performance enhancement with respect to bandwidth utilization and content availability, there are intrinsic drawbacks in both schemes which render them infeasible to be incorporated into BitTorrent. The first coding scheme, i.e., the one using all plain blocks to generate coded blocks, incurs a very high coding complexity and disk operation overhead, and the potential of improving content availability in the second coding scheme cannot be fully exploited due to the slow generation rate of new blocks.

To show the complexity of computation and the overhead of disk operations involved in the first coding scheme, consider a peer intends to split a k bytes file into n blocks for distribution. Since a coded block is a linear combination of all n plain blocks, this peer should read k bytes from disk and perform k multiplications and $\frac{(n-1)k}{n}$ additions for each uploading; hence this peer should read at least nk bytes and perform nk multiplications and $(n-1)k$ additions to render content available. While in BitTorrent, this peer needs to read only k bytes and perform no arithmetic calculations. Moreover, the decoding process is also a computation-intensive operation. When a peer has downloaded n linearly independent coded blocks, in order to recover the original file, it should first compute the inverse of the coding matrix, with a time complexity of $O(n^3)$, and then perform nk multiplications and $(n-1)k$ additions. In BitTorrent, no computation is needed in order to recover the original file. As a result, when the distributed file is very large, e.g., with a size of 1 GB, the overhead incurred by this coding scheme is not acceptable for most users.

The second network coding scheme postulates a high heterogeneity of blocks. In

other words, the number of different coded blocks should be kept at a high level. In order to achieve a high block heterogeneity, the generation of new blocks must be fast. In the second coding scheme, however, since a coded block consists of only two plain blocks, a peer can generate a new block only after it has downloaded at least two coded blocks, each sharing a common underlying plain block with another one. If a peer has downloaded $\frac{n}{2}$ blocks, each with two underlying plain blocks not contained in other $\frac{n}{2}-1$ blocks, then it has to download one more block in order to generate the first new coded block. Thus, in the worst case, a peer may make no contribution to the improvement of block heterogeneity in half of its download time. More importantly, as a peer can generate only one new coded block from two coded blocks which share one common underlying plain block, many coded blocks generated at different peers are identical, which significantly restricts the block heterogeneity. Although the first problem may be addressed by the block request policy which preferentially requests blocks from which new coded blocks can be generated, the second situation is unavoidable.

In order to accelerate the generation of new blocks and thus improve the heterogeneity of blocks, we propose another simple but effective sparse network coding scheme. Unlike the aforementioned second network coding scheme, in which a coded block is generated from two random plain blocks, we stipulate that a plain block can only be combined with another fixed plain block for generating coded blocks, and in this way, any two coded blocks either have the same two underlying plain blocks or share no common plain blocks. The rationale behind using two blocks for generating new coded blocks is that a peer only needs to download two coded blocks with the same underlying plain blocks in order to generate new coded blocks, which leads to a high generation rate of new coded blocks.

To specifically describe how the proposed network coding scheme works, suppose a peer intends to distribute a file with n plain blocks, each with a size of k bytes. With a little abuse of notations, we represent these n plain blocks by a set $S = \{b_1, b_2, \ldots, b_n\}$.

This peer then generates n coded blocks in the following way. It first groups the n plain blocks into $\frac{n}{2}$ disjoint sets, each with two plain blocks that are not included in other sets. For each set $s = \{b_i, b_j\}$, this peer selects two linearly independent vectors, $c_i = [c_{i1}, c_{i2}]$ and $c_j = [c_{j1}, c_{j2}]$, and in both vectors, all elements are non-zero and selected from a finite field. Then two new blocks are generated by:

$$\begin{cases} b'_i = c_{i1}b_i + c_{i2}b_j \\ b'_j = c_{j1}b_i + c_{j2}b_j \end{cases} \tag{4.8}$$

In this way, the n coded blocks generated by this peer can be used to reconstruct the original file.

Once a peer has downloaded two coded blocks, denoted by b'_i and b'_j, with the same underlying plain blocks, it can use b'_i and b'_j to recover the two underlying plain blocks, and further generate a new coded block from b'_i and b'_j. Since the two coding coefficients of each coded block are randomly selected from a finite field, the probability that the coding coefficient vectors of two coded blocks are linearly dependent is $\frac{1}{q-1}$, which implies that any two coded blocks which are generated by different peers and have the same underlying plain blocks can be decoded with a probability of $\frac{q-2}{q-1}$.

We now give an analysis on the overhead of the proposed coding scheme. Since a peer can generate a new block only when it has downloaded two blocks with the same underlying plain blocks, the number of new blocks generated at this peer is $\frac{n}{2}$, and thus this peer needs to read n blocks from the disk and perform nk multiplications and $\frac{nk}{2}$ additions. Moreover, the plain blocks can be obtained by solving a linear system of equations with the same structure as equation (4.8), which has the same computation complexity with the generation of a new block. In order to recover the original file, one only needs to solve $\frac{n}{2}$ such linear systems of equations. Therefore, the proposed coding scheme incurs an overhead with a complexity of $O(nk)$, and this complexity is much less than that of the overhead incurred in the first coding scheme, which is

$O(n^2k)$. Moreover, the overhead of the proposed coding scheme can be further reduced by applying lazy coding [76]. By using lazy coding, a peer creates a virtual code block after downloading two coded blocks with the same underlying blocks, and generates the corresponding real code block upon request. In this way, the blocks which have not been requested during a peer's lifetime are not generated, which reduces both computation and disk read operations.

We then analyze the probability of a peer with i blocks is of no use to another peer with j blocks, i.e., $\Pr(A_{i,j})$, under the proposed coding scheme. It is easy to see that $\Pr(A_{i,j})$ takes the largest value when the peer with j blocks can recover j plain blocks, i.e., each of j blocks has the same underlying plain blocks with another one. In this case, $\Pr(A_{i,j}) = \frac{\binom{j}{i}}{\binom{n}{i}}$, which is equal to that in original BitTorrent swarms. Therefore, the usability of peers is enhanced with the proposed network coding scheme.

The proposed coding scheme is also able to improve the content availability. Without loss of generality, we stipulate that under the proposed coding scheme, the plain block b_{2i-1} can only be combined with block b_{2i} to generate new blocks. For each plain block b_i in a swarm without network coding, we replace it with b'_i in the following way:

$$b'_i = \begin{cases} c_1 b_{i-1} + c_2 b_i, & \text{if } i \bmod 2 = 0 \\ c_1 b_i + c_2 b_{i+1}, & \text{otherwise} \end{cases}$$

Furthermore, in the replacement, if a peer owns both b_{2i} and b_{2i-1}, the two coded blocks used to replace these two plain blocks must have linearly independent coding coefficient vectors.

We now demonstrate how the probability of content being available varies as a result of replacement. Consider two plain blocks b_{2i} and b_{2i-1}. If both blocks reside at the same peer before the replacement, the availability of them remains the same after the replacement. If there is no peer owning both blocks before the replacement, the probability of both blocks being available before replacement is $\max\{0, 1 - 2^{1-t}\}$,

where t is the number of peers owning either b_{2i} or b_{2i-1}, and after the replacement, this probability will become $\max\{0, 1 - q^{1-t}\}$, where q is the size of the finite field. Therefore, it can be easily shown that the probability of content being available after the replacement is no smaller than that before replacement, and we can thus conclude that although the proposed coding scheme is very simple, it can improve the availability of content.

4.4 Block Scheduling Algorithm

In order to integrate the proposed coding scheme with BitTorrent, we need to modify the block scheduling policy of BitTorrent as the original policy does not support network coding. Since the original *rarest-first* block scheduling policy of BitTorrent exhibits a high bandwidth utilization, in the hope of reserving this nice feature, we propose a variant of the original rarest-first policy, which computes the importance of blocks according to their underlying plain blocks. The proposed block scheduling algorithm is shown in Algorithm 1.

The main function of Algorithm 1 is to determine which block located at a specific neighbor is to be requested. Given the block availability information of a neighbor id, the local peer assigns a weight to each useful block located at id, and then selects a block with the least weight for request. The weight of a block b is determined according to the distribution of the blocks with the same underlying plain blocks as b among all neighbors. Specifically, in the execution of the algorithm, each time a block with the same underlying plain blocks as b is found, the weight of b will be increased by a value which is determined according to whether the new found block and b are linearly independent or not. The weight of b will be increased by β if the new found block and b are linearly independent, and α otherwise. We also stipulate that $\beta < \alpha$ since we want to give preference to blocks which have less linearly dependent counterparts in other neighbors.

Algorithm 1: Block scheduling algorithm

Input: Block availability information at each neighbor, and the neighbor to which the local peer will send request, denoted by id

```
foreach coded block i at id do
    weight_i = 0;
    if this block is useful to local peer then
        foreach neighbor n_j do
            if there is a coded block k at n_j with the same underlying plain
            blocks as i then
                if there exists a non-zero c such that i = ck then
                    weight_i = weight_i + α;
                end
                else
                    weight_i = weight_i + β;
                end
            end
        end
    end
end
```

Randomly select a block with the smallest non-zero weight, i.e., an element of the set $\{r|0 < weight_r \leq weight_{r'}, \forall r' : weight_{r'} > 0\}$;

Send a request for this block to id;

To see the reason why $\beta < \alpha$, consider a scenario that there are two blocks, b and b', with the same underlying plain blocks at neighbor id, both of which are useful to the local peer, and in other neighbors, there is only one block b'' which has the same underlying plain blocks as b and satisfies $b'' = ib$, where i is non-zero and selected from the finite field. In this case, it is more appropriate to request b' than to request b as b and b'' are actually the same and thus a more copy of b' would result in the identical number of copies of b and b'. In the proposed algorithm, the existence of b'' would result in the weight of b and b' being increased by α and β respectively, and thus in order to assign a higher priority to b', β must be smaller than α since the block with smallest weight would be requested, as specified in the algorithm.

We then discuss the time complexity of Algorithm 1. In the proposed coding scheme, the n plain blocks are grouped into $\frac{n}{2}$ sets. Without loss of generality, we consider a

coding scheme in which the $2i$-th plain block can only be combined with $(2i+1)$-th plain block to generate a coded block. Thus each peer is able to store the downloaded coded blocks in an array in the way that a coded block generated from j-th plain block and $(j+1)$-th plain block is stored at the j-th or $(j+1)$-th position.[1] Under this coding scheme, the second "if" statement can be carried out in constant time, and the third "if" statement can also be carried out in constant time as it is only a determination of the linear dependence of two 2-dimension vectors. Moreover, the first "if" statement is a combination of the second and third "if" statements. Therefore, all three "if" statements can be carried out in constant time, and thus the time complexity of Algorithm 1 is $O(mn)$, where m is the number of neighbors and n is the number of coded blocks at neighbor id.

4.5 Performance Evaluation

In this section, computer simulations are performed to evaluate the performance of our proposed scheme. In our experiments, four BitTorrent implementations, which are 1) the original BitTorrent, 2) BitTorrent with adaptive neighbor selection [44], 3) BitTorrent with FEC and 4) BitTorrent with the proposed coding scheme, are compared with respect to the amount of control messages, content availability and bandwidth utilization[2]. For simplicity, the four implementations are respectively represented by "Pure BT", "Adap BT", "FEC", and "BT+NC" in the figures of the following sections. The simulation results reveal that the proposed coding scheme is very effective and greatly outperforms the other three implementations with regard to content availability

[1] For any other approach of block combination, we can renumber the plain blocks such that the $2i$-th plain block is combined with $(2i+1)$-th plain block after renumbering, and when all the plain blocks have been decoded, the original file can be reconstructed by reversely renumbering the plain blocks. In this regard, all approaches of block combination are equivalent.

[2] The reason we choose the second and third implementations for comparison purposes is that they can both enhance the performance of BitTorrent swarm in some specific aspect, as we will show. Besides, we also have considered the modifications proposed in [79, 56]. However, as we mentioned before, if all peers in the swarm run the same BitTorrent variation (either [79] or [56]), the overall bandwidth utilization will be lower than that of peers running the original BitTorrent clients [56, 20].

at a cost of a very slight increase in the control overhead.

4.5.1 Experimental Setup

As a large peer population and a long swarm lifetime could more accurately reveal the essential characteristics of the swarms, we deploy multiple swarms where a file with a size of 100MB is distributed, and 5000 peers with a homogeneous bandwidth of 50KB/s participate in the download. Peers arrive the swarm at a rate of λ, according to a Poisson process, and abort the download at a rate of θ. The residence time of *publisher* [1] is 5 hours, which is enough for the swarm entering the steady-state, and all the other seeds immediately leave the swarm as soon as they complete the download. All the seeds, including the publisher, would not return to the swarm after departure. Hence there is no seed after the departure of the publisher. The size of a block is set to $128KB$, and thus the total number of the plain blocks in the swarm is 800. The other parameters used in the simulation are the default value in BitTorrent settings. The variables in Algorithm 1, i.e., α and β, are set to 2 and 1, respectively, and the impact of the different values of α and β is discussed in Section 4.5.6. The finite field used in simulations is GF(256).

In the FEC implementation of BitTorrent, besides the 800 plain blocks, the publisher also generates 50 additional blocks, each being a linear combination of all plain blocks, such that any 800 blocks can be used to reconstruct the original file. The reason why the number of additional blocks is set to 50 is twofold. First, as explained in Section 4.3, the overhead of computation and disk operation incurred in the process of reconstruction grow quickly as the increase of the number of additional blocks, which has a negative impact on the feasibility. Second, the increase of the number of additional blocks may reduce the opportunity of a block being requested, which in turn decreases the survival time of blocks.

[1] we use *publisher* to represent the seed initiating the swarm.

4.5.2 Control Overhead

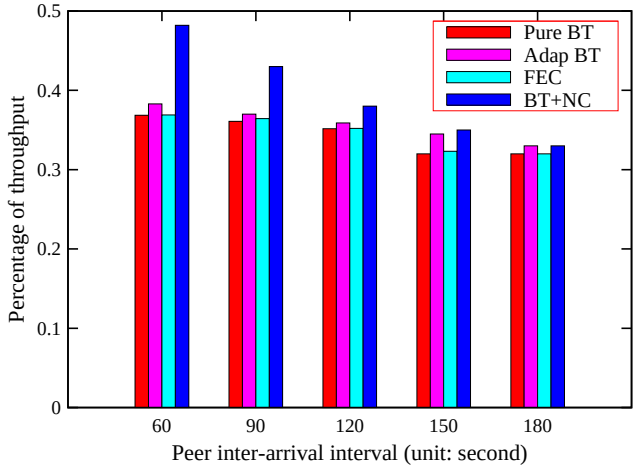


Figure 4.1: The percentage of throughput composed of control overhead in different implementations of BitTorrent

Fig 4.1 describes the control overhead incurred by different implementations of BitTorrent. In all four implementations, the control overhead is mainly caused by the announcement of the availability of new downloaded or generated blocks. From this figure we can see that although the proposed coding scheme incurs a slight higher control overhead than the other three implementations do, it is still practically feasible as the amount of control overhead is very limited, only accounting for less than 0.5% of the total throughput. The reason to the higher overhead of the proposed coding scheme is that in other three implementations, a peer needs to only tell its neighbors the *id* of the newly downloaded block in order to announce the availability of this block, while in the implementation with the proposed coding scheme, the coding coefficients of the two plain blocks also have to be transmitted in order to announce the availability of a coded block. As the reduction of peer arrival rate, the control overhead incurred by the proposed coding scheme experiences a moderate decrease, and meanwhile narrows the difference between itself and the overhead incurred by other implementations. This is due to that the decrease of peer arrival rate reduces the number of neighbors of each

peer, and thus decreases the amount of transmitted control messages.

4.5.3 Content Availability

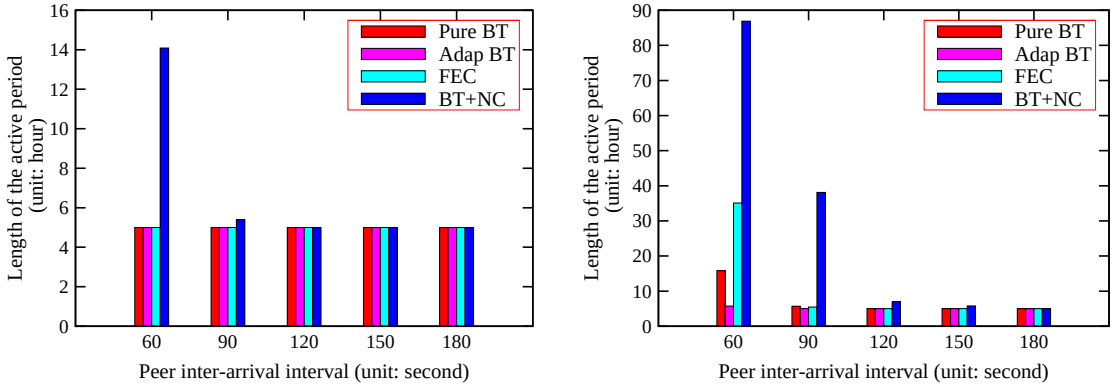
Let $R(t)$ denote the dimension of the space spanned by the set of coding coefficient vectors of the blocks existing in the swarm at time t. If $R(t) = n$, where n is the number of plain blocks of the original file, then the content is available at time t, since the original file can be recovered from the existing coding blocks.

Then we intend to show the length of the period during which the availability of content equals 1 since it can decide the service capability of the swarm, i.e., the number of peers able to complete the download. For simplicity of expression, we use "active period"[1] to represent the period during which the content is available. The longer the active period is, the more peers the swarm can serve. Since in the simulations all the seeds would never return after departure, the active period is actually the interval between the time that the swarm is initiated and the first time that *Rank* falls below n. Suppose the publisher initiates the swarm at time 0. Then the length of the active period of this swarm, denoted by LAP, is

$$LAP = \inf\{t | R(t) < n\} \tag{4.9}$$

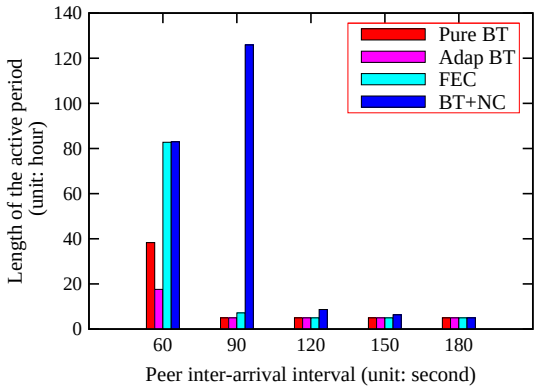
The lengths of the active periods in different BitTorrent implementations are shown in Fig. 4.2. From Fig. 4.2 it is easy to see that, with respect to the length of the active period, the implementation with the proposed coding scheme performs better if not the same as the other three implementations in all scenarios. When $1/\theta = 1800s$, the proposed coding scheme could increase the length of the active period by about 10 hours, provided $1/\lambda = 60s$, and for other peer arrival rates, all four implementations exhibit the same performance regarding the length of the active period. In the cases

[1]The "active period" defined here is in essence the same as the counterpart defined in Chapter 3.

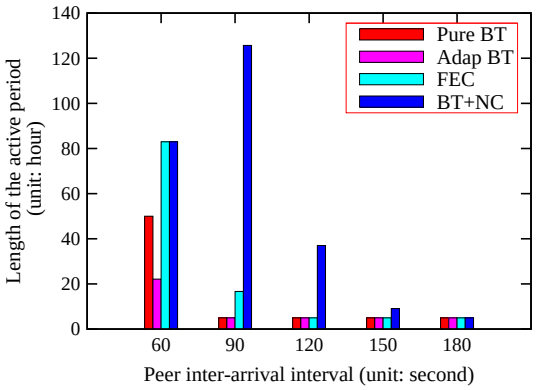


(a) $1/\theta = 1800s$

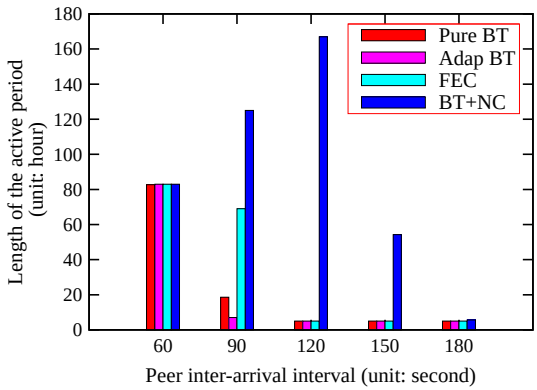
(b) $1/\theta = 3600s$



(c) $1/\theta = 5400s$



(d) $1/\theta = 7200s$



(e) $1/\theta = \infty$

Figure 4.2: The lengths of the active periods in different implementations of BitTorrent

that $1/\theta \geq 3600s$, the content keeps available during the whole lifetime of the swarms[1] with the proposed coding scheme, provided that the peer inter-arrival interval equals $60s$. In contrast, in order to reserve the availability of content during the lifetime of the swarms with FEC implementation when $1/\lambda = 60s$, it should be satisfied that $1/\theta \geq 5400s$, while under the same condition, the content is always available in the swarms with the proposed coding scheme as long as $1/\lambda \leq 90s$. Moreover, if peers never abort the download, i.e., $1/\theta = \infty$, in the swarms with $1/\lambda \leq 120s$, the proposed coding scheme is able to render the content available all the time, and in the swarm with $1/\lambda = 150s$, the proposed coding scheme enables the content to remain available for a long time, namely 50 hours, after the departure of the publisher. By contrast, the content in the swarms with the other three BitTorrent implementations becomes unavailable as soon as the publisher departs in the cases that $1/\lambda \geq 120s$.

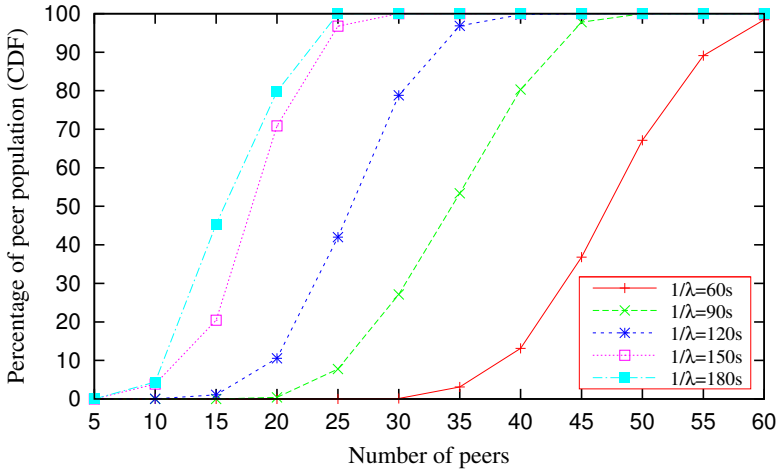


Figure 4.3: The CDF of peer population

We then show how many peers in the swarm are necessary in order to render content available by investigating the distribution of peer population before the content being

[1]The lifetime of a swarm is roughly the product of the peer inter-arrival interval and the number of peers in the swarm. Since the number of peers in each swarm in 5000, the lifetimes of the swarms with the peer inter-arrival interval being $60s$, $90s$, $120s$, $150s$ and $180s$ are roughly 83 hours, 125 hours, 167 hours, 208 hours and 250 hours, respectively.

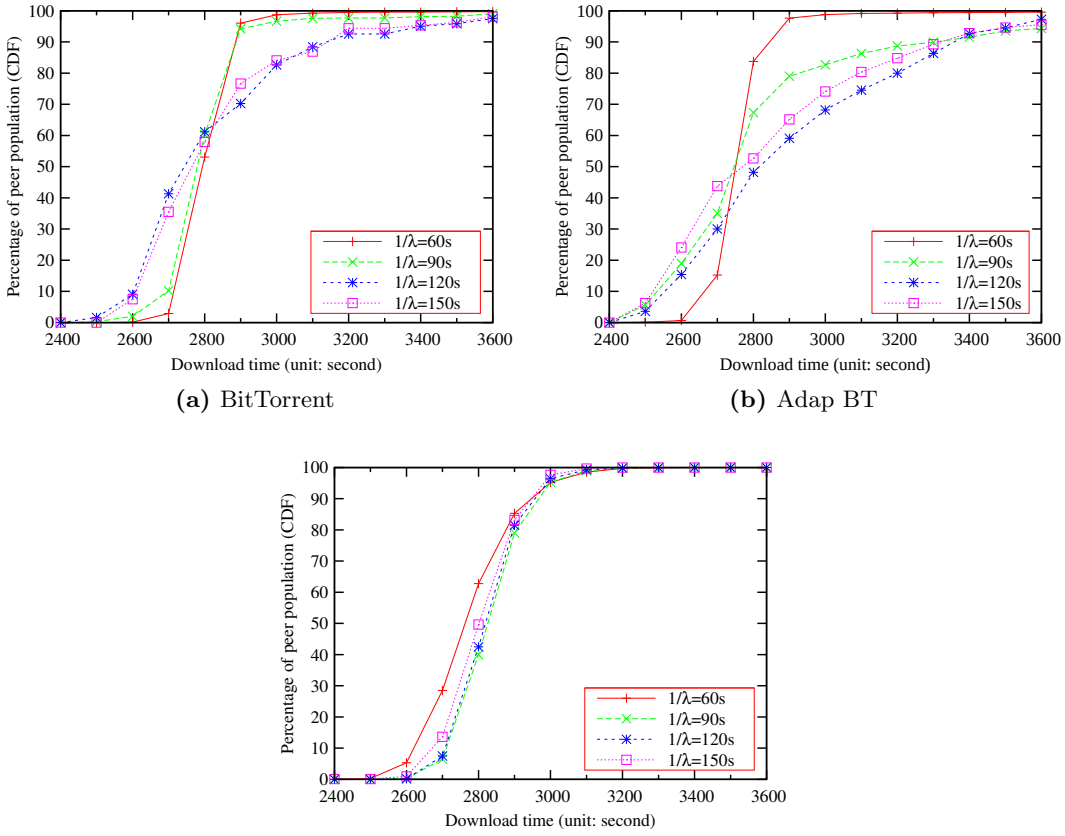
unavailable. Fig. 4.3 illustrates the distributions of peer population in the swarms with no peer aborting the download before completion. Since the proposed coding scheme has little effect on the probability distribution of peer population in the swarm[1], only the probability distribution of peer population in the swarms with the proposed coding scheme is shown in Fig. 4.3. As shown in Fig. 4.2, in the original BitTorrent swarm, the content soon becomes unavailable after the departure of the publisher when $\frac{1}{\lambda} \geq 90s$, and it can thus be inferred that without the presence of seeds, the content tends to be unavailable in the original BitTorrent swarms when the number of peers in the swarm is less than 40. In contrast, If the proposed coding scheme is integrated with BitTorrent, the content is highly likely to be available even there are only less than 15 peers in the swarm, as Fig. 4.2 demonstrates that even when $\frac{1}{\lambda} = 150s$, the content remains available in nearly 50 hours after the departure of the publisher. Therefore, the proposed coding scheme is very effective in regard to improving the availability of content in the swarm with an unpopular file distributed, i.e., a low peer arrival rate.

4.5.4 Bandwidth Utilization

We then discuss bandwidth utilization in different implementations of BitTorrent. The distributions of download time under different peer arrival rates, denoted by λ, are given in Fig. 4.4. The download time in FEC implementations is not provided since it has almost the same distribution as peer download time in original BitTorrent. From this figure we can obtain that when $1/\lambda = 60s$, 28% of peers complete the download in $2700s$ in the swarm with the proposed coding scheme, while in the BitTorrent swarm, peers with the download time less than $2700s$ only account for about 3 percents of peer population. However, under the same peer arrival rate, the average download time of peers in swarm with the proposed coding scheme is very close to that of peers in

[1]As we will show in next section, the download time keeps nearly unchanged when using the proposed coding scheme as BitTorrent itself is very effective with regard to bandwidth utilization, and thus the peer population in the swarm is also unchanged when using the proposed network coding scheme.

BitTorrent swarm, and the "Adap BT", which exhibits the best performance among all the implementations, only decreases the average download time by less than 1 minutes, as shown in Fig. 4.5. This is because the BitTorrent protocol itself is very effective in respect of bandwidth utilization, and thus there is little space for improvement [13].



(a) BitTorrent

(b) Adap BT

(c) BT with proposed coding scheme

Figure 4.4: Distributions of download time under different peer arrival rates

In Fig. 4.4, only the download times of peers who complete the download are gathered, and the number of peers completing the download is shown in Fig. 4.6. Nevertheless, when taking into consideration the peers which have not completed the download at the time content becomes unavailable, we can say that the proposed coding scheme could indeed reduce the download time of peers. Specifically, when $1/\lambda \geq$

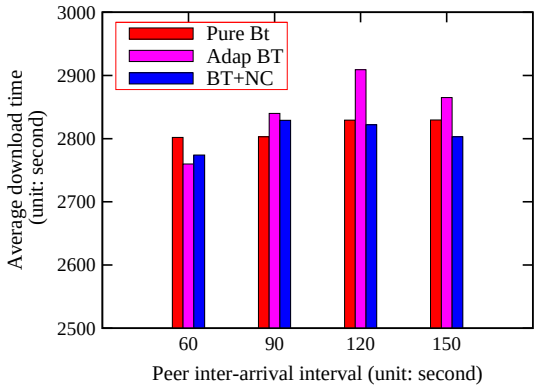


Figure 4.5: Average download time of peers

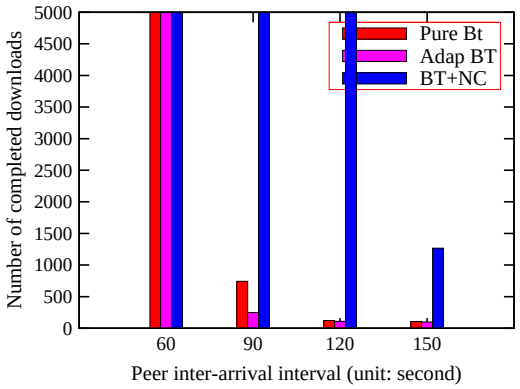
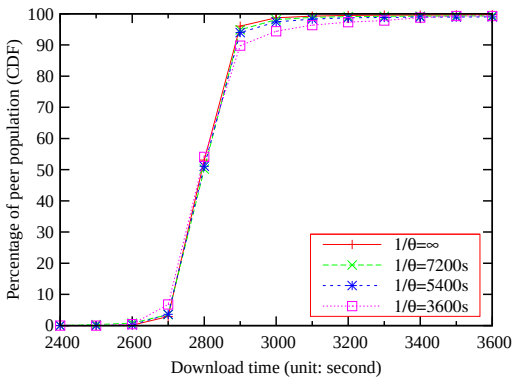
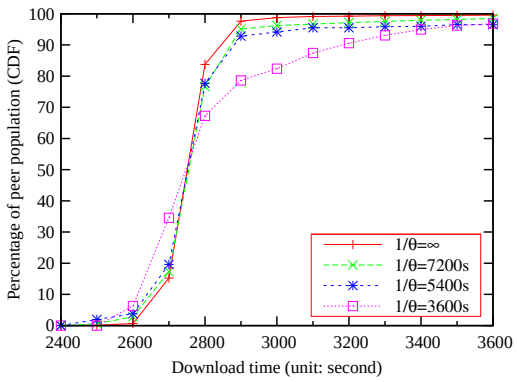


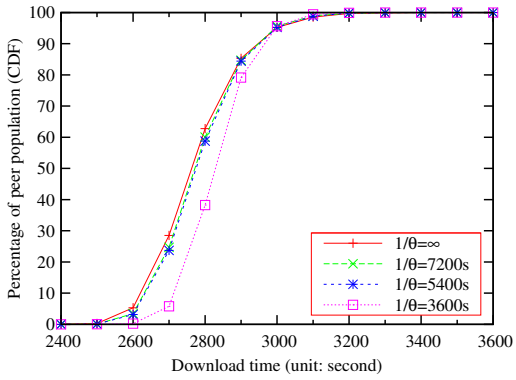
Figure 4.6: Number of peers completing the download



(a) BitTorrent



(b) Adap BT



(c) BT with proposed coding scheme

Figure 4.7: Distribution of peers' download time under different download abortion rates

$90s$, as shown in Fig. 4.2, the active period of swarms with the proposed coding scheme is much longer than the active period of swarms with the other three BitTorrent implementations, which means that the number of peers able to complete the download experiences a great increase when the proposed coding scheme is adopted, as shown in Fig. 4.6. Since after the content becomes unavailable, many peers may choose to wait for a seed to supplement the lost blocks, our coding scheme could eliminate or significantly decrease this waiting time by prolonging the length of the active period and thereby enabling more peers to complete the download before content becomes unavailable. In this regard, the download time of peers is decreased by the proposed coding scheme.

We can also draw from this figure that in the swarm with the proposed coding scheme, the increase of peer inter-arrival interval, $1/\lambda$, from $60s$ to $90s$, slightly increases the peer download time. The reason to the increase of download time is that, as we will show in Fig. 4.8, the generation rate of new blocks slows down due to the increase of $1/\lambda$. The continual increase of $1/\lambda$, however, has little effect on the download time. In the swarms with the other two BitTorrent implementations, it can be seen that the increase of peer inter-arrival interval leads to an increase in the variance of peer download time.[1]

We also investigate the effect of peers aborting the download on the download time of peers. Fig. 4.7 illustrates how peers' abortion of download affects the distribution of the download time of peers which complete the download in swarms with $1/\lambda = 60s$. As in Fig. 4.4, the distribution of peer download time in FEC implementation is also not provided in this figure since it is roughly the same as that in original BitTorrent. From this figure, we can see that the peers' abortion of download hardly affects the

[1] As shown in Fig. 4.2, the active periods in BitTorrent swarms with $1/\lambda \geq 1.2s$ last only 5 hours, and thus there are roughly 18000λ peers able to complete the download in these swarms, which may lead to the slight difference between the distribution of peer download time in these swarms and the distribution in BitTorrent swarms with $1/\lambda \leq 90s$. For the same reason, we do not plot the distribution of peer download time in both implementations with $1/\lambda = 180s$.

peer download time in BitTorrent swarm, but incurs a slight increase in the average download time of "Adap BT" peers. For the swarm with the proposed coding scheme, the distribution of peer download time keeps almost unchanged when $1/\theta$ varies from ∞ to $5400s$. This can be attributed to the high block diversity in the corresponding swarms: once a peer aborts the download, its neighbors could always find alternatives from which they can download the desired blocks. When $1/\theta$ decreases to $3600s$, the download time experiences a slight increase, and this is probably because the comparatively high download abortion rate may lead to a degraded block diversity, which renders it not so easy for peers to find the alternative to the blocks of failed neighbors from other neighbors.

4.5.5 Decoding Process

In the proposed coding scheme, two coded blocks consisting of the same plain blocks can be used to reconstruct the two corresponding plain blocks and generate a new coded block. Consequently, each peer would generate $n/2$ new coded blocks in the download process, and it makes more contribution to improving block diversity to generate new coded blocks earlier. In this section, we examine the generation of new blocks under different peer inter-arrival intervals, denoted by $1/\lambda$, and show that new coded blocks are generated at a high rate, which provides a sound evidence to the improvement of content availability induced by the proposed coding scheme.

In the simulations, the number of plain blocks is 800 since the file for distribution has a size of 100MB and the block size is 128KB. We explore how many blocks have been downloaded before a constant number of new coded blocks being generated at each peer and plot its distribution in Fig. 4.8. The line labeled by "Top-c" corresponds to the distribution of the number of blocks that have been downloaded before the first c new coded blocks being generated. From this figure we can see that the generation of new blocks is relatively slow at the start of the download: there are roughly 50 blocks that have been downloaded before the generation of the 10-th new block when $1/\lambda = 60s$,

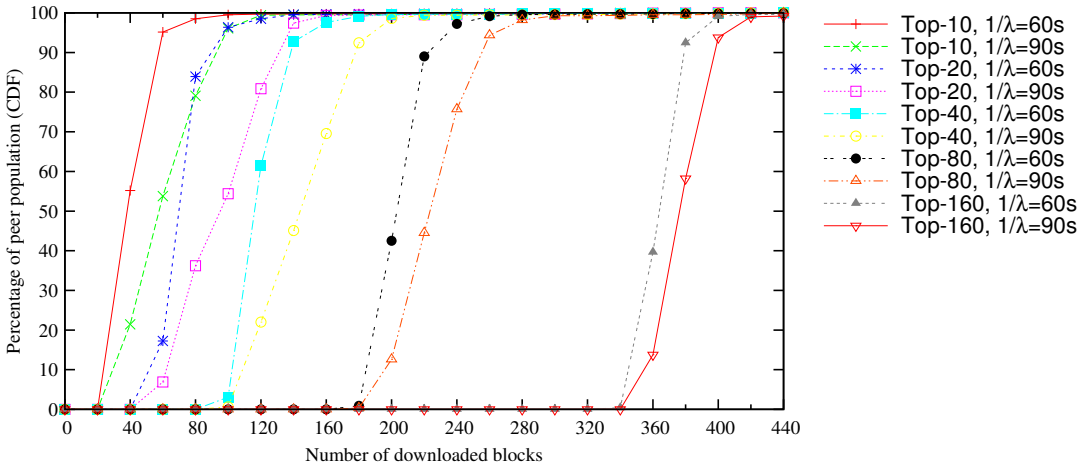


Figure 4.8: The generation rate of new blocks under different peer inter-arrival intervals

and this number goes to about 60 when the peer inter-arrival interval increases to 90 seconds. This is easy to understand since when a peer has only a few blocks, it is less likely for this peer to hold blocks with the same underlying plain blocks. As the download proceeds, the generation of new blocks also accelerates: most peers have generated more than 80 new blocks when they complete the download of the first 240 blocks, and when a peer has downloaded 380 blocks, it is almost certain that this peer has already generated more than 160 new blocks. Therefore, we come to a conclusion that although the proposed coding scheme stipulates that only two fixed plain blocks can be combined, this stipulation exerts little negative impact on the diversity of blocks.

The increase of peer inter-arrival interval has a negative effect of the generation of new blocks. As shown in Fig 4.8, before the generation of a constant number of new blocks, a peer in the swarm with $1/\lambda = 90s$ downloads about 20 more blocks than a peer in the swarm with $1/\lambda = 60s$ does. This is also easy to understand since the increase of peer inter-arrival interval would lead to the reduction in the number of online peers, which in turn slows down the generation of new blocks. The deceleration of the generation of new blocks also explains the increase of peer download time shown in Fig. 4.4c. We also obtain from the figure that this negative effect will weaken as more blocks

are downloaded. Moreover, as we observe from the simulations, when $1/\lambda \geq 90s$, the continual increase of peer inter-arrival interval exerts litter impact on the generation of new blocks, which also coincides with that the peer download time keeps nearly unchanged when $1/\lambda$ varies from $90s$ to $150s$, as can be observed in Fig. 4.4c. For this reason, we do not plot in Fig. 4.8 the distribution for swarms with $1/\lambda \geq 120s$.

4.5.6 Different values of α and β

The preceding simulations are run under the settings that α and β equal to 2 and 1 respectively. In this section, we explore whether different values of α and β influence the performance of the proposed coding scheme. The lengths of the active periods and the average download times when α and β take different values are demonstrated in Fig. 4.9 and 4.10.

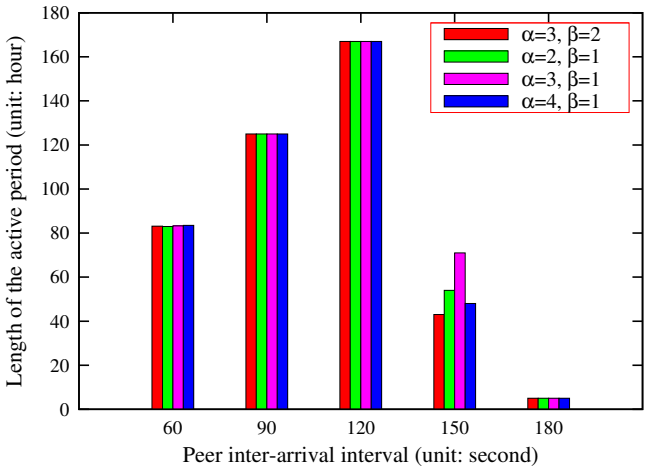


Figure 4.9: The impact of different values of α and β on the lengths of the active periods

From Fig. 4.9, we can obtain that concerning the length of the active period, the swarms with different values of α and β perform a little differently in the case that the peer inter-arrival interval, i.e., $1/\lambda$, is $150s$, while in other cases that the peer inter-arrival interval is less than or equal to $120s$, different values of α and β exert no influence on the length of the active period. Meanwhile, it can be observed from Fig.

4.10 that the average download time keeps unchanged as the variation of the ratio of α to β. Therefore, we can conclude that the different values of α and β have little impact on the proposed coding scheme, with respect to the length of the active periods and the average download time.

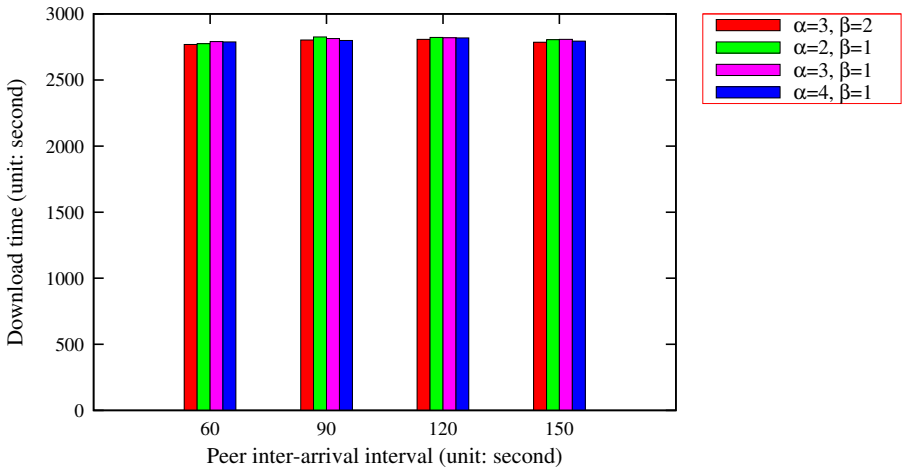


Figure 4.10: The impact of different values of α and β on the average download times

The explanation to the above observations is that the new coded blocks generated at peers will always have a higher priority to be requested than the old ones, as long as $\alpha > \beta$. When a peer generates a new coded block b, it is unlikely that there exists a block b' in the swarm such that $b = cb'$, where c is a non-zero element selected from the finite field. As a result, when one neighbor of this peer, A, executes Algorithm 1, if block b is useful to peer A, this block will be assigned with a weight of $m\beta$, where m is the number of blocks which have the same underlying plain blocks as b and are owned by the neighbors of peer A. Since

$$m\beta \leq m_1\alpha + m_2\beta,$$

$$\forall \alpha, m_1, m_2 : \alpha > \beta, 0 \leq m_1, 0 \leq m_2, m_1 + m_2 = m$$

the block b has the highest priority to be requested among all the m blocks. Conse-

quently, the new generated coded blocks can be disseminated quickly after birth, which not only contributes to the block diversity, but also improves the bandwidth utilization of the peers generating these blocks.

4.6 Conclusion and Discussion

Although there are numerous BitTorrent swarms deployed in BitTorrent, many of them suffer from content unavailability. We aim to investigate the feasibility of using network coding to improve the availability of content in this chapter. We first analyze the impact of network coding on the availability of content and the usability of a peer to other peers, and show that network coding has the potential to ameliorate the availability of content of BitTorrent swarms.

In order to leverage this potential of network coding and keep the overhead incurred at a low level, a simple sparse network coding scheme is proposed, in which a plain block can only be combined with another specific plain block to generate a new coded block. Since only two plain blocks are involved in the generation of new blocks, the computation complexity and disk operation overhead in the coding process are kept at a low level, and new blocks can be generated at a peer when it has downloaded two coded blocks with the same underlying plain blocks, which implies that the generation rate of new blocks is very fast and thus the block heterogeneity in the swarm is improved. A block scheduling algorithm based on the BitTorrent's built-in *rarest-first* policy is presented to adapt BitTorrent to the proposed coding scheme.

The effectiveness of the proposed coding scheme on the availability of content in the BitTorrent swarms has been demonstrated through extensive simulations and performance comparisons. With the proposed network coding scheme, the duration of period during which no seed is present but content is available is greatly prolonged, and only a few peers are enough to render the content available.

The proposed coding scheme may also find its application in peer-assisted online

hosting systems [63] [95]. Since with the proposed coding scheme, only a few peers are enough to render the content available in a swarm, the server could reduce the bandwidth allocated to this swarm, and uses the saved bandwidth resource to serve more swarms. In such a way, both the service capability of the server and the service quality perceived by peers are enhanced.

Chapter 5

A Detailed Survey on a Large Private BitTorrent Community

5.1 Introduction

As we identified in Section 2.4, there have been several studies on private BitTorrent communities [102, 22, 73, 46]. However, the behaviors of users in private BitTorrent have not been examined comprehensively in the existing studies , and there is lack of an analysis on the inter-relationships among different respects of user behavior.

This chapter presents a thorough survey on one of the largest private BitTorrent community, CHDBits [9]. We study the torrent and user behavior from a variety of aspects, and present an in-depth analysis of user behavior, thereby bridging the gap existing in the literature [102, 22, 73, 46]. The main contributions of this study are summarized as follows.

We first present a detailed analysis of the torrents of CHDBits. We find that although there are a great number of torrents with a very long age, 95 percent of torrents have at least one seed, showing a high content availability in CHDBits. By studying the distribution of seeders and leechers, we show that most torrents have a high ratio of seeders to leechers. We also investigate the download rate of users in each torrent, and obtain that most users could download at a satisfactory rate. We then study the user behavior from various perspectives. First, we present the distribution of the number of users in different levels. By investigating the variation of the number of active users in a ten-day period, we infer that a great number of users participate in downloading and uploading as soon as they can. We also study the number of completed downloads for each user, and observe that most users are very enthusiastic

for downloading: 90 percent of users have completed the downloads of more than 16 torrents. By exploring the upload and download traffic of each user, we show that, as a result of participation in many torrents, most users have a huge amount of both upload and download traffic, and maintain a share ratio above 1. The seeding and leeching time of each user are also inspected, and it is found that almost all users have demonstrated a great willingness to serve leechers after the completion of download, and thus achieve an extremely high ratio of seeding time to leeching time.

We finally present an in-depth analysis of user behavior by investigating the influence of user age and bandwidth on users' participation in torrents, on user traffic and on seeding and leeching time. We observe that compared with higher bandwidth users, lower bandwidth users more often participated in the torrents with smaller content sizes, and are more likely to participate in high-popularity torrents. We also obtain that higher bandwidth users tend to participate in more torrents, and hence generate a larger amount of traffic than lower bandwidth users. However, we find that the seeding time of high bandwidth users does not correspond to number of torrents they have participated in: compared with low bandwidth users, many high bandwidth users with the similar age have spent much less time in seeding, although they may have participated in more torrents.

The remainder of this chapter is organized as follows. In Section 5.2, we give an introduction to CHDBits. The methodology of data collection is given in Section 5.3. We present the survey result and the corresponding analysis in Section 5.4, and finally conclude this chapter in Section 5.5.

5.2 CHDBits

CHDBits [9] is one of the most popular and largest private BitTorrent communities in China. On 2011/09/07, there are more than 38,000 torrents in CHDBits, and this number is still increasing, as each day some new torrents would be uploaded by CHDBits

users. Since most of contents distributed in CHDBits are HD movies, the corresponding files of torrents are likely to have a large size, and by 2011/09/07, the aggregate size of files distributed among community users has already exceeded 360 TB. In addition, CHDBits has more than 30,000 registered users, and has contributed almost 100 PB of Internet traffic by 2011/09/07.

Fig. 5.1 demonstrates the variation of traffic in CHDBits during the period 2011/08/28-2011/09/06. It can be observed from this figure that CHDBits generates roughly 214 TB of upload traffic per day, which means the average aggregate upload rate of the community exceeds 20 Gbits/s. We can also obtain from this figure that the overall download traffic is about one quarter of the aggregate upload traffic, and this is because the download traffics of many torrents are counted at a discount, and moreover, for many popular torrents, in a specific time period after their releases, the download traffic is not counted at all.

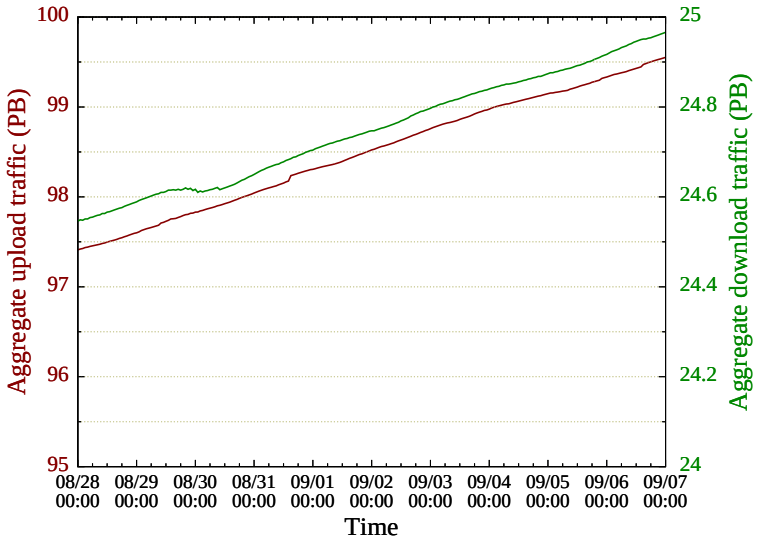


Figure 5.1: The variation of traffic in CHDBits during the period 2011/08/28-2011/09/06

Users in CHDBits are classified according to the download traffic, share ratio and registration time, and different privileges are granted to users in different levels. The details of the user hierarchy in CHDBits are shown in Table 5.1.

user level	requirements
Peasant	$d \geq$ 10GB and $r \leq 0.6$
	$d \geq$ 50GB and $r \leq 0.7$
	$d \geq$ 100GB and $r \leq 0.8$
	$d \geq$ 200GB and $r \leq 0.9$
	$d \geq$ 400GB and $r \leq 1.0$
User	new registered user
Power user	$d \geq$ 50GB, $r \geq 3.05$, and $t \geq 5$ weeks
Elite user	$d \geq$ 120GB, $r \geq 3.55$, and $t \geq 10$ weeks
Crazy User	$d \geq$ 300GB, $r \geq 4.05$, and $t \geq 15$ weeks
Insane user	$d \geq$ 500GB, $r \geq 4.55$, and $t \geq 20$ weeks
Veteran user	$d \geq$ 750GB, $r \geq 5.05$, and $t \geq 25$ weeks
Extreme user	$d \geq$ 1TB, $r \geq 5.55$, and $t \geq 25$ weeks
Ultimate user	$d \geq$ 1.5TB, $r \geq 6.05$, and $t \geq 30$ weeks
Nexus master	$d \geq$ 3TB, $r \geq 6.55$, and $t \geq 30$ weeks

Table 5.1: User hierarchy in CHDBits. The symbols d, r and t in this table represent the download traffic, share ratio and the time that a user has been registered, respectively.

5.3 Survey methodology

Elaborate information of torrents and users of CHDBits can be directly obtained from the torrent publish site. For each torrent, CHDBits records the size of the corresponding content, age, number of both seeders and leechers, and more importantly, the detailed information of each completed download corresponding to this torrent, known as a snatch in BitTorrent community, is also available. In addition, for each user, CHDBits preserves the accumulative upload and download traffic, as well as the accumulative seeding and leeching time.

The comprehensive information of torrents and users available in CHDBits significantly facilitates our survey. We crawl the CHDBits website to retrieve the aforementioned information and store it into a database to simplify the analysis. The data collection is performed multiple times during the period 2011/08/03/-2011/09/07, and each time we were able to collect more than 5,500,000 download records, which manifests a high level of participation of CHDBits users in downloading and uploading.

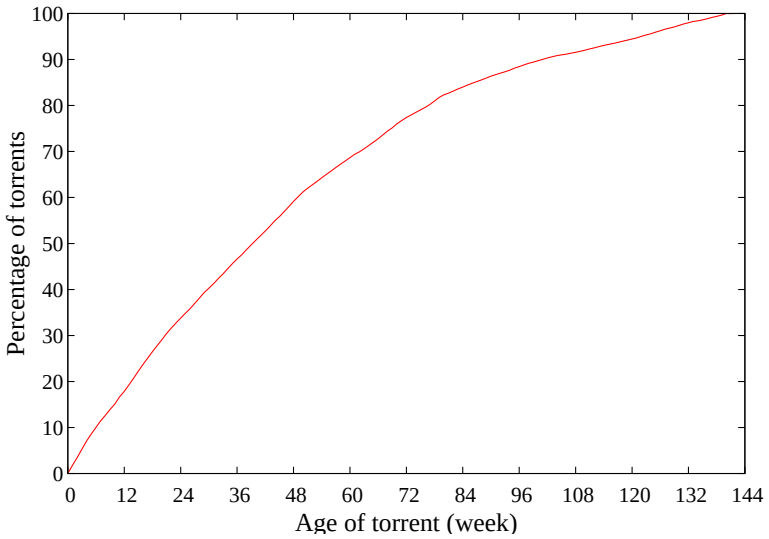


Figure 5.2: The distribution of torrent age in CHDBits (CDF)

5.4 Survey results and analysis

5.4.1 Torrent

The torrents in CHDBits are studied in the respects of torrent age, torrent size and popularity.

5.4.1.1 Torrent evolution

As mentioned before, in CHDBits, there are more than 38,000 torrents, and each day there are some new torrents added by users. The cumulative distribution function (CDF) of torrent age in the snapshot 2011/09/03[1] is shown in Fig. 5.2, from which we can obtain that a large fraction of torrents have a very long age, e.g., almost half of torrents have been alive for more than 40 weeks. Another observation of this figure is that the growth rate of the number of torrents becomes higher with time, and this is mainly due to the increasing user number of CHDBits.

[1]If not otherwise stated, the data in the snapshot of CHDBits collected on 2011/09/03 will be used in the figures presenting the status of CHDBits at a specific time. The selection of snapshot time exerts little impact on system status, since CHDBits has run for a long time.

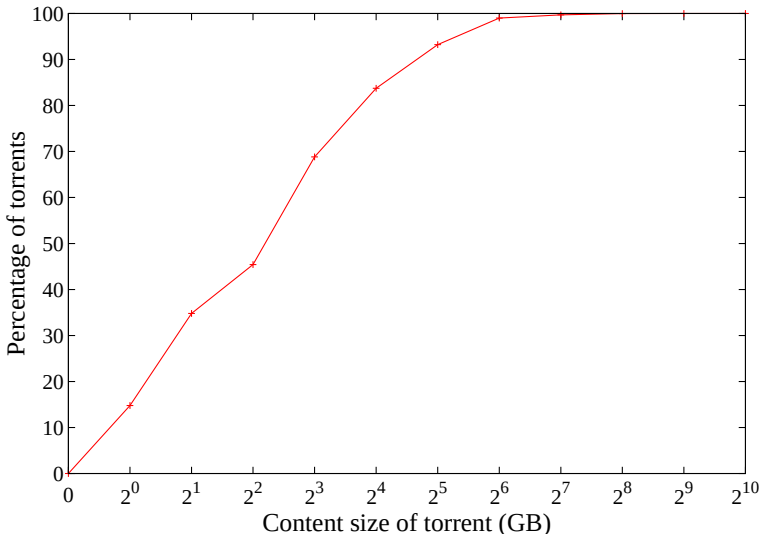


Figure 5.3: The distribution of the content size in CHDBits (CDF)

5.4.1.2 Distribution of torrent content size

The aggregate torrent content[1] size of CHDBits is a little more than 360 TB on 2011/09/03, and the distribution of content size of each torrent in snapshot 2011/09/03 is demonstrated in Fig. 5.3. Since there are many high definition movies in CHDBits, many torrents have a large content size. As shown in this figure, while the contents of only about 15 percent of torrents are less than or equal to 1GB, there are more than half of torrents with content size larger than 4 GB, and 18 percent of torrents have a content size larger than 16 GB. Moreover, torrents with content size larger than 64 GB compose 1 percent of total torrent population. Besides, we also notice that the top 10 percent of torrents, with regard to the content size, account for almost half of the aggregate content size.

5.4.1.3 Torrent popularity and content availability

We also investigate the popularity of the torrents by examining the number of snatches, seeders and leechers in each torrent. As stated before, a snatch represents a completed

[1]The content of a torrent is the distributed file corresponding to this torrent.

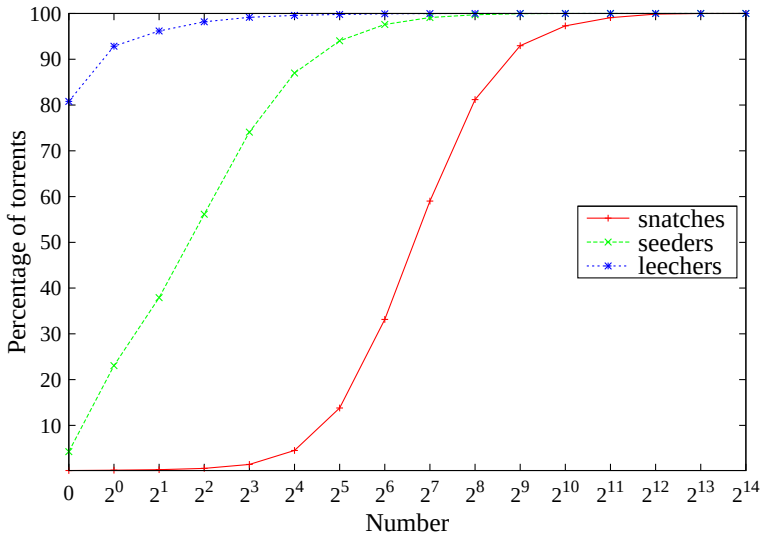


Figure 5.4: The distribution of the population of snatches, of seeders and of leechers in CHDBits (CDF)

download of the corresponding torrent content. Thus the number of snatches of a torrent can to a large degree reflect the popularity of this torrent. As demonstrated in Fig. 5.4, the content of more than 95 percent of torrents has been downloaded for more than 16 times, and the total number of the torrents, with the corresponding content being downloaded for more than 512 times, is 2,745, which is 7 percent of the torrent population. Again, the top 10 percent of torrents, regarding the number of snatches, account for almost 50 percent of the completed downloads, although not very obvious in the figure.

The distributed contents in CHDBits are highly available. As can be observed from Fig. 5.4, more than 95 percent of torrents have at least one seeder. Taking into consideration that almost half of the torrents were released 40 weeks ago, it is effortless to conclude that for most torrents, the release time has little influence on the availability of the corresponding content.

Compared with the seeders, leechers are rather sparsely distributed. Fig. 5.4 shows that only about 20 percent of torrents have leechers, and the torrents with more than 4

leechers only make up 0.85 percent of torrent population. This is partly because most users are only interested in the new torrents, and pay little attention to the old ones [40]. Another reason is that as implied in the distribution of the number of snatches, many users only focus on a small part of torrents, and the other torrents are thus unfrequented. The sparse distribution of leechers leads to a high ratio of seeders to leechers in most torrents, and the downloading users, i.e., leechers, can benefit from this situation since they have multiple data sources and thus be able to download at a high rate.

5.4.1.4 Download rate of users in each torrent

Fig. 5.5 depicts the average download rate of users in each torrent. In this figure, the torrents are numbered in the increasing order of the average user download rate. It can be observed that there are about 12,500 torrents with the average user download rate less than 256 Kbytes/s, out of which 50 percent have an average user download rate less than 128 Kbytes/s. There are roughly the same number (10,500) of torrents with the average user download rate in the range from 256 Kbytes/s to 512 Kbytes/s as there are with the average user download rate in the range from 512 Kbytes/s to 1 Mbytes/s. In addition, the average download rates of users in more than 5,300 torrents are higher than 1Mbytes/s. We can thus be able to conclude that users in most torrents could download at a satisfactory rate.

The small part of torrents, which have a low average user download rate, may correspond to those with an unstable *publisher*[1]. If the publisher becomes offline for many times before all the content of the torrent has been uploaded to others, the average user download rate of the corresponding torrent will be prolonged accordingly.

[1]The user who uploads a torrent to CHDBits and serves as the original seed for this torrent is called the *publisher* of this torrent

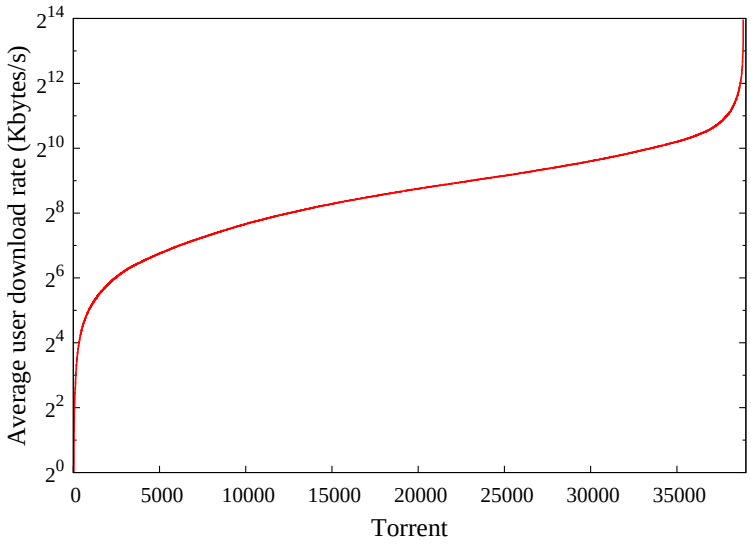


Figure 5.5: The average download rate of users in each torrent

5.4.2 User

Since a part of users prevent their profiles from being accessed from other users, we were not able to collect the data of all users. Fortunately, these users only form 18 percent of the total population, and the profiles of the rest 82 percent users, namely, about 24,600 users, are still available.

5.4.2.1 User level distribution

The distribution of user level in snapshot 2011/09/03 is given in Fig. 5.6, in which ten user levels listed in Table 5.1 are numbered from 1 to 10 in the increasing order of download traffic requirement, with peasant being the level 1 user and nexus master being the level 10 user. We can obtain from Fig. 5.6 that there are most users in level 3 and level 4: each level has more than 7,000 users, and accounts for roughly a quarter of the total population. As the requirement of user level promotion becomes more stringent, from the fourth level, there are less users in higher levels, opposite to the variation trend of user number in the first four levels. Fig. 5.6 also implies that the majority of users maintain a satisfactory share ratio as only less than 200 users are

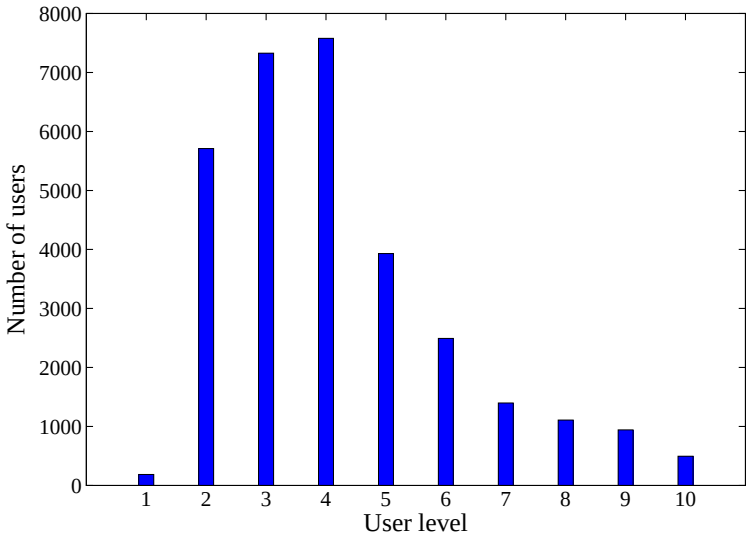


Figure 5.6: User level distribution

peasants.

5.4.2.2 Seeders, leechers and active users

The degree of participation of CHDBits users in uploading and downloading can be reflected by the variations of the numbers of seeders, leechers and active users in CHDBits. A user is active if it is being involved in at least one torrent, as either a seeder or a leecher. Since users may participate in multiple torrents, an active user may thus correspond to multiple seeders and/or leechers. Fig. 5.7 shows how the numbers of seeders, leechers and active users varied during a ten-day period (2011/08/28-2011/09/06). As shown in this figure, all the three numbers demonstrated nearly the same diurnal pattern during the period, and the two ratios, namely, the ratio of seeders to leechers and the ratio of seeders to active users, were thus stable during the whole period, being about 18 and 27, respectively.

Another observation of this figure is that the variation of the three numbers during the weekdays[1] is different from that during the weekends. On weekdays, all three numbers first experience an obvious decrease from 0:00 to 7:00, as this is the sleep

[1]During the ten-day period, 08/28, 09/03, and 09/04 are weekend.

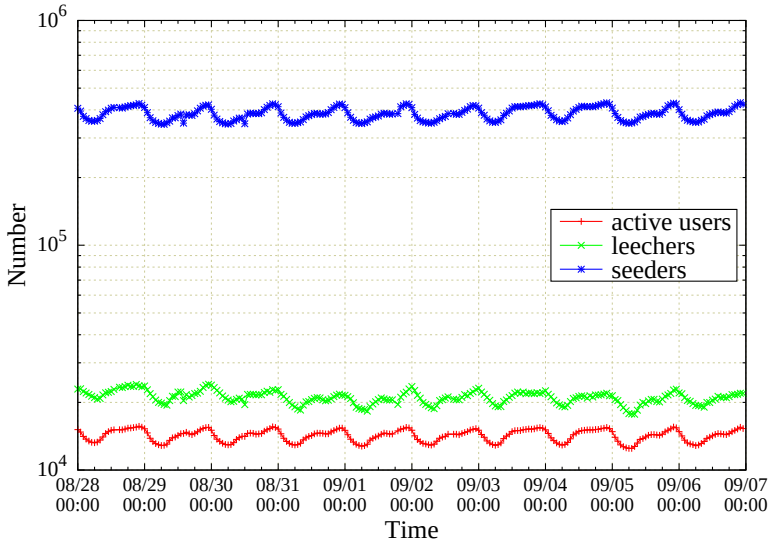


Figure 5.7: The variation of the number of seeders, leechers and active users during 2011/08/28-2011/09/06

time for most users. However, even at 7:00, there are still about 13,000 active users. Starting from 7:00, the numbers begin arising steadily until 12:00, and then keep almost unchanged in the following 6 hours. The three numbers experience another increase from 18:00, which corresponds to the time that people come back from work, and arrive at their maximum values at 24:00. At weekends, the variations of the numbers before 8:00 are roughly the same as what happens in the same time period during weekdays. However, at weekends, the increase rates of the numbers during 8:00-12:00 are much higher than they are on weekdays, and all three numbers attain their maximum values at 12:00, in contrast to 24:00 on weekdays, and then stay at that level during the rest of the day. Therefore, it can be inferred that many users, who participate in seeding and/or leeching during 18:00-24:00 on weekdays, become active during 8:00-12:00 at weekends, showing their great enthusiasm for uploading and downloading.

5.4.2.3 The download history of users

For each user, we count the number of snatches in which it is involved, and then plot the distribution of this number in Fig. 5.8. It can be observed from the figure that

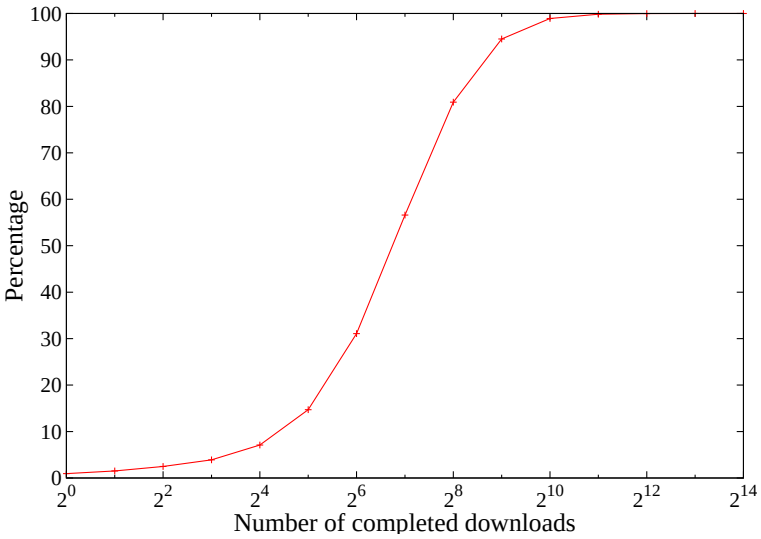


Figure 5.8: The distribution of the number of the completed downloads of each user (CDF)

there are only about 10 percent of users with the number of completed downloads less than or equal to 16. This is because these users are mainly those registered the account in recent time, and it is highly likely that the number of completed downloads of these users would increase with time. Among the rest 90 percent of users, more than 40 percent were involved in more than 128 snatches, and users with the number of completed downloads larger than 256 make up 20 percent of the total population. In addition, there exists a user with the number of completed downloads larger than 8192, and we find from the survey data that this user has already completed the download of more than 12,000 torrents, and the daily average number of completed downloads of this user is about 12. Moreover, since each snatch corresponds to one completed download, the number of snatches in which a user is involved might be less than the number of torrents this user has participated in as this user may abort some downloads before completion.

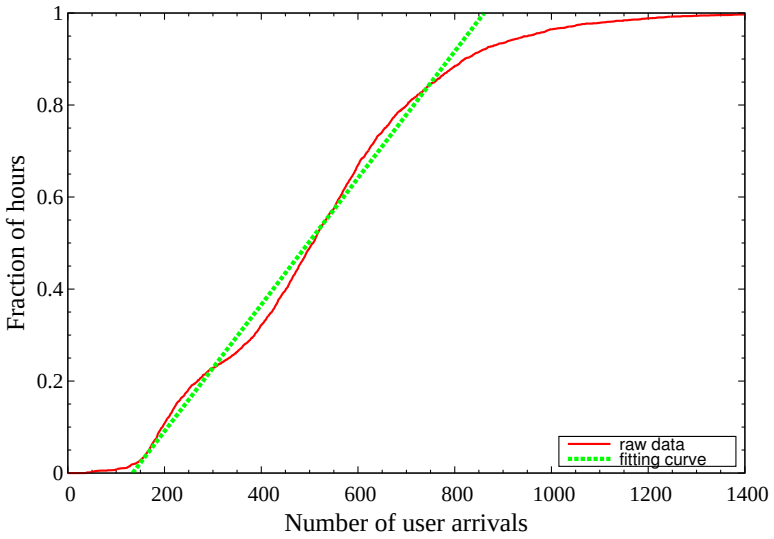


Figure 5.9: Distribution of the number of user arrivals per hour (CDF)

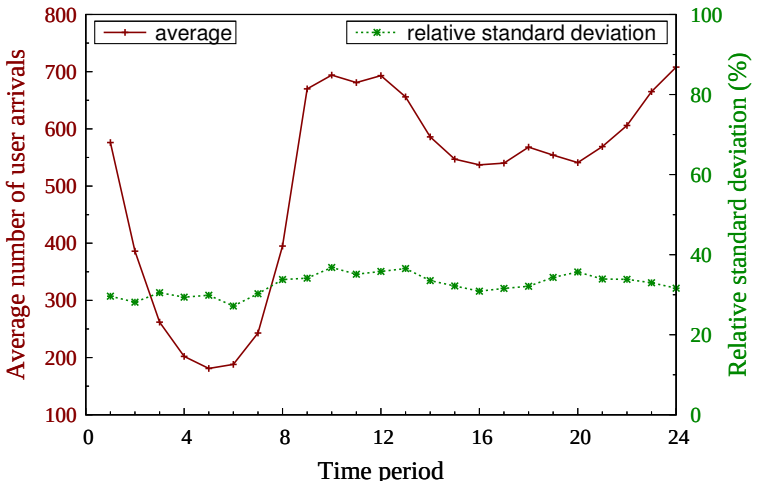


Figure 5.10: The average and the relative standard deviation of the number of user arrivals in different time periods

5.4.2.4 User arrival pattern

Figure 5.9 shows the cumulative distribution function of the number of hourly user arrivals. It can be shown from this figure that in most (more than 80 percent) hours during the five-month period, the number of user arrivals is almost uniformly distributed in the range between 175 and 750, and there are more than 750 users arriving in the remainder hours.

In order to show the relationship between time and user arrival rate, we divide a day into 24 time periods, with the i-th time period representing the i-th hour of a day, and calculate the average number of arrivals in each time period. The calculation result is presented in Figure 5.10. It can be obtained that the number of user arrivals in each time period remains relatively steady, as the relative standard deviation of the number of user arrivals in each time period is about 30%. In addition, we can also observe that the user arrival rate dovetails well into users' daily schedule. For example, In the first six time periods of a day, the number of user arrivals experienced a significantly decrease as these time periods are the sleep time for most users, and in the following four time periods, the number of user arrivals undergoes a rapid increase with the increase rate similar to the decrease rate of user arrivals in the sleep time. In addition, it can be obtained from this figure that CHDBits users arrive at a higher rate in the morning than they do in the afternoon and evening.

We now investigate the arrival pattern of individual users. During the investigation, we exclude the 2,071,132 records of completed download tasks with unavailable user information, and from the remainder records, we identify 23,500 users, which form the basis of the investigation. Figure 5.11 shows the interval between two consecutive arrivals of an identified user. As shown in this figure, the probability that an user arrival took place in the same day as the last arrival of the same user did is about 0.55. We can thus infer that if one day a user issued a new download task, it is with more than half possibility that this user would launch one or more download tasks in the

same day; in other words, individual users arrive in a burst way.

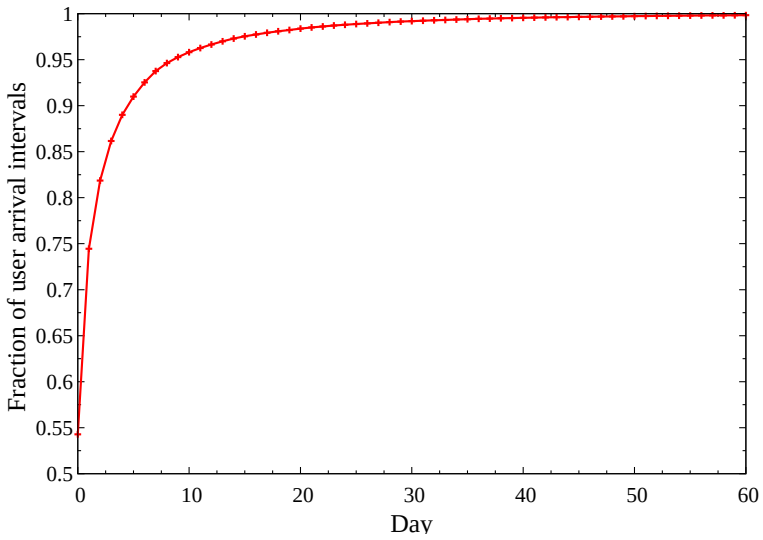


Figure 5.11: Distribution of the interval between two consecutive arrivals of a single user (CDF)

Figure 5.12 describes the probability distribution of the number of daily arrivals per user, and it is found in this figure that a user issues no download task in a day with a probability of 0.766636. Using this probability distribution, we can estimate how many days are needed for all CHDBits users to issue at least one download task. Let X_i represent the number of days passed until the next arrival of the i-th user, and assume that all X_i have the same probability distribution. It can thus be easy to see that X_i is a geometrical random variable with the following probability density function:

$$\mathbb{P}\{X_i = k\} = (1-p)^{k-1}p, \quad k = 1, 2, \ldots,$$

where $p = 0.233364$, as a user issues no download task in a day with a probability of about 0.766636. Let M_n denote the number of days for n users to issue at least one download task, and we thus have $M_n = \max\{X_1, X_2, \ldots, X_n\}$. According to [34], the

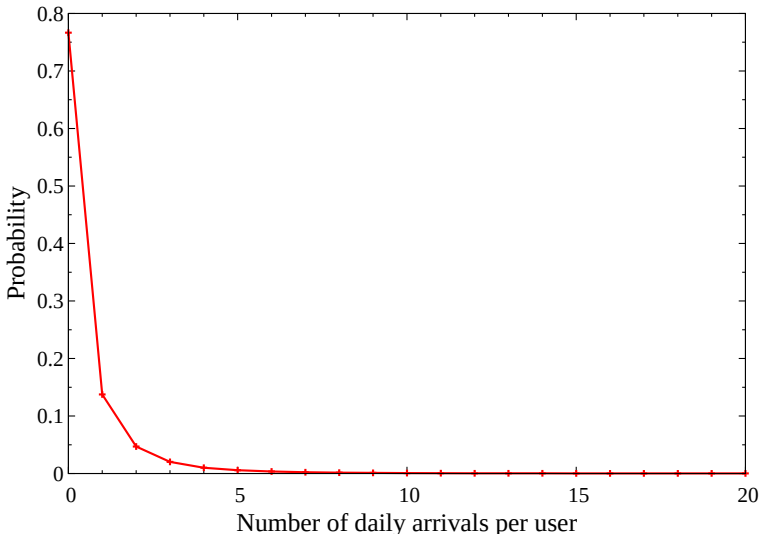


Figure 5.12: Probability distribution of the number of daily arrivals per user

expectation of M_n satisfies the following condition:

$$-\frac{1}{\log(1-p)}\sum_{k=1}^{n}\frac{1}{k} < \mathbb{E}M_n < 1 - \frac{1}{\log(1-p)}\sum_{k=1}^{n}\frac{1}{k} \tag{5.1}$$

As we mentioned before, there were 31,399 users registered with CHDBits in 2011/11/16. By substituting n and p in equation (5.1) by 31,399[1] and 0.233364 respectively, we can obtain that $41.1366 < \mathbb{E}M_n < 42.1366$, which means all CHDBits users are expected to launch at least one download task in about 42 days.

5.4.2.5 User traffic

The user level distribution cannot accurately reflect the upload and download traffic of users, since many users with a great amount of upload traffic also belong to low levels, which occurs when their download traffics do not achieve the requirement for level promotion, or are too large such that the share ratio requirement is not satisfied.

[1]This number will increase as new users are invited to join CHDBits; however, the increasing rate is rather slow [18]. Moreover, as can be inferred from equation (5.1), increasing n by 1 only results in an increase of $-\frac{\log^{-1}(1-p)}{n+1}$ in $\mathbb{E}M_n$, and thus for large n, the increase of n has only a very limited influence on the expectation of M_n.

It is thus necessary to independently investigate the distribution of the upload and download traffic of users.

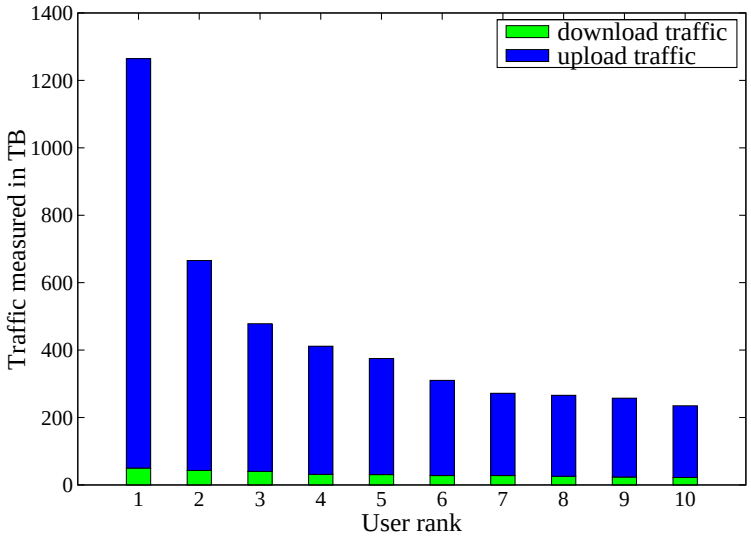


Figure 5.13: Top 10 upload users and top 10 download users

Fig. 5.13 shows the top 10 upload users and top 10 download users. It is obvious that the user with the largest upload traffic has already contributed roughly 1.2 PB of data, which makes up more than 1 percent of total traffic generated at CHDBits[1], and the aggregate upload traffic contributed by top 10 upload users exceeds 4 PB.

Compared with the upload traffics of the top 10 upload users, the download traffics of the top 10 download users are far less. The user ranking first with respect to download traffic has downloaded about 50 TB of data, and the download traffics of the top 10 download users amount to about 320 TB, which is only roughly 8 percent of the aggregate upload traffic of the top 10 upload users. However, considering that the overall content size of all torrents is a little more than 350 TB, we can say that these users are very active in joining the download. Moreover, as the download traffics of many torrents are counted at a discount, which we have mentioned above, the real

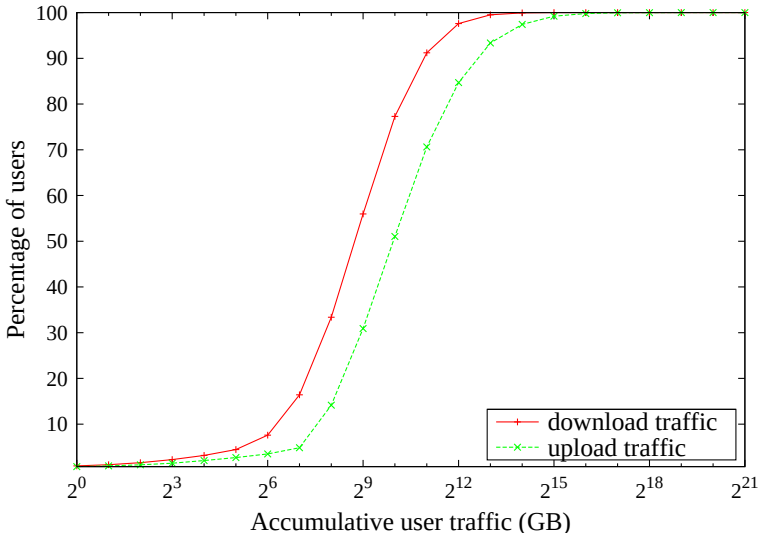
[1]Recall that the total traffic generated by CHDBits users was a little less than 100 PB on 2011/09/03, as shown in Fig. 5.1.

download traffics of the top 10 download users may be much more than that exhibited in Fig. 5.13. For instance, we notice that the user ranking second with respect to download traffic has completed the download of the contents of more than 12,000 torrents, which correspond to a real download traffic of 77.45 TB.

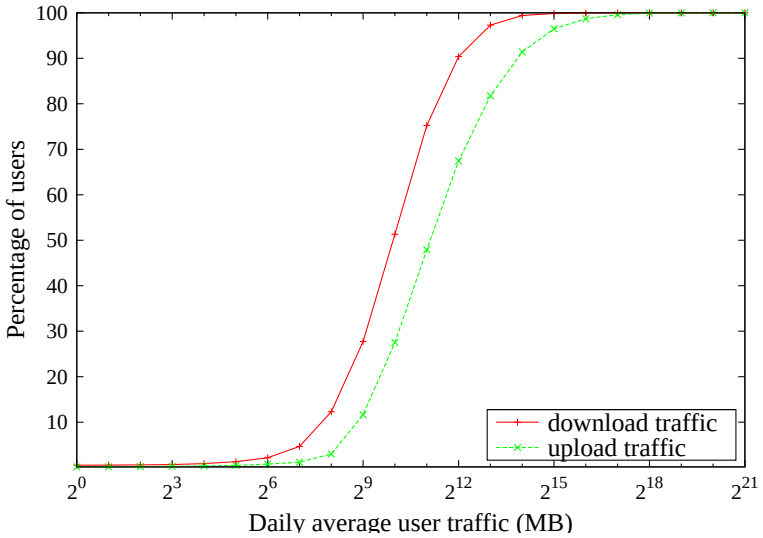
Fig. 5.14a shows the distribution of accumulative upload and download traffic of users. It can be inferred from Fig. 5.14a that most users have a large download traffic: more than 90 percent of users have downloaded more than 64 GB of data, and users with the download traffic more than 1 TB compose more than 20 percent of the total population. Compared with the download traffic, many users have an even larger upload traffic. More than 90 percent of users have contributed more than 128 GB of upload traffic, and there are about half of users with an upload traffic more than 1 TB, out of which 14 percent have uploaded more than 8 TB of data. In addition, We can also find that users with more than t GB of download traffic constitute roughly the same fraction of the total population as users with the upload traffic more than $2t$ GB do.

There are also a small part of users with far less download or upload traffic, and this may have several reasons. These users may have their accounts registered lately, and thus there is little time for them to improve their traffic statistics. In addition, they may have a low network bandwidth, which renders earning traffic difficult, and the third reason is that these users may be inactive in downloading and uploading.

We also explore the average traffic a user generated in each day, and plot the corresponding distribution in Fig. 5.14b. It is shown in Fig. 5.14b that more than 95 percent of users have either a daily average download traffic of more than 128 MB, or a daily average upload traffic of more than 256 MB. About half of users generate a download traffic more than 1 GB each day on average, and by contrast, there are 70 percent of users with a daily average upload traffic of more than 1 GB. The user with the largest daily average upload traffic contributes more than 1 TB of data per



(a) The distribution of accumulative user traffic (CDF)



(b) The distribution of daily average user traffic (CDF)

Figure 5.14: The distribution of the upload and download traffic of each user

day on average, and we can thus infer that the bandwidth of this user is at least 12 Mbytes/s. We also notice in Fig. 5.14b that there are roughly the same number of users with the daily average download traffic more than t MB as there are with the daily average upload traffic of more than $2t$ MB do, the same observation as pointed out in Fig. 5.14a.

An interesting observation can be derived by comparing Fig. 5.14a with Fig. 5.14b. Let U, u, D and d denote accumulative upload traffic, daily average upload traffic, accumulative download traffic and daily average download traffic of each user, respectively, and all the four traffic metrics are measured in MB. Then for each integer i, we have

$$|\Pr(d \leq 2^i) - \Pr(D \leq 2^{i+9})| \leq 0.04$$
$$|\Pr(u \leq 2^i) - \Pr(U \leq 2^{i+9})| \leq 0.06$$

In other words, u and d have the similar probability distribution to $U/2^9$ and $D/2^9$, respectively.

5.4.2.6 User time

We then investigate the degree of users' participation in downloading and uploading by examining the seeding and leeching time of each user. Fig. 5.15 shows the distributions of accumulative user time and daily average user time, in the same way as how user traffic is shown in Fig. 5.14. It can be shown from Fig. 5.15a that users have spent much more time in seeding than in leeching. There are 5 percent of users with the leeching time less than 1 week, and more than half of users spending less than 16 weeks in downloading. In sharp contrast, almost all users have seeded for at least one week, and users with the seeding time longer than 16 weeks constitute roughly 90 percent of population. In addition, there are more than 10 percent of users with the seeding time longer than 1024 weeks, in contrast to only 3 thousandths of users who have leeched

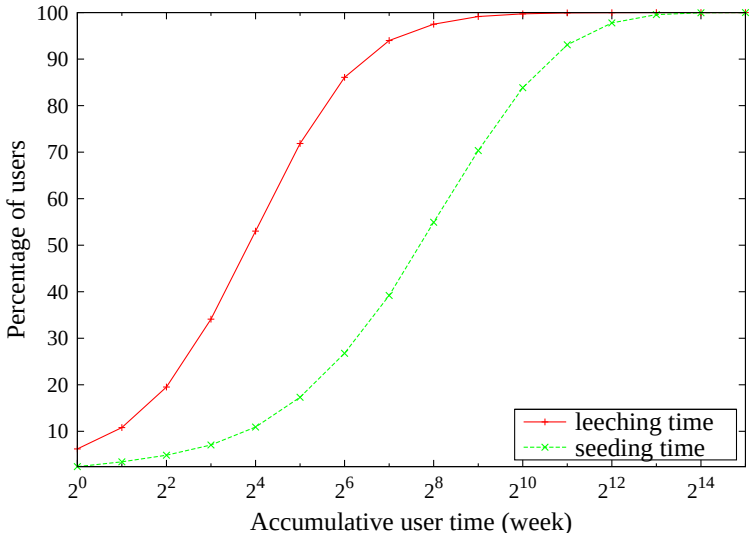
for more than 1024 weeks.

We also study how much time a user spends in seeding and leeching each day, and show the corresponding distribution in Fig. 5.15b. It can be shown that while more than 10 percent of users spend an average time less than or equal to 1 hour in downloading each day, there are few users with the daily average seeding time less than 1 hour. Furthermore, 80 percent of users spend less than 16 hours in downloading each day on average, and by contrast, there are about the same fraction of users with the daily average seeding time longer than 16 hours. In addition, more than 20 percent of users have a daily average seeding time longer than 256 hours, which means that these uses have to seed for almost 24 hours each day, provided that the number of seeding torrents is no more than 11.

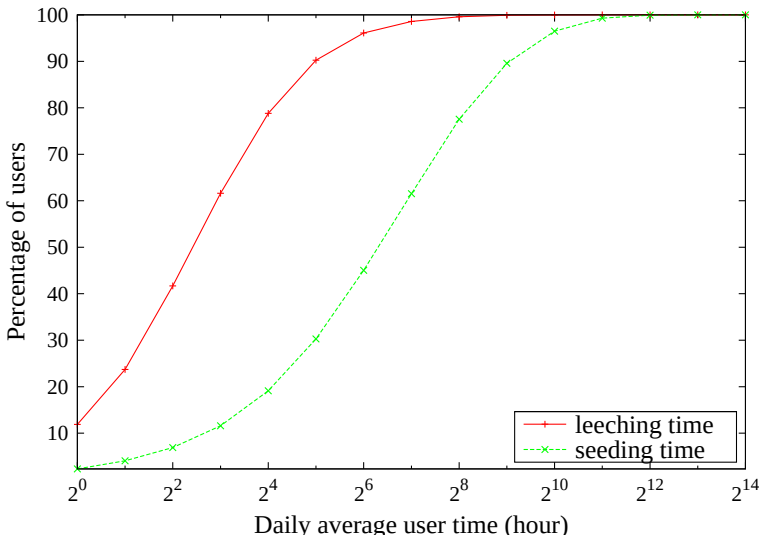
The seeding strategies of users are also investigated. Fig. 5.16 describes per torrent seeding time for each user. In this figure, we exclude users with the number of completed downloads less than 10, as we intend to get more samples for each user. The number of rest users is 23,413, implying only less than 5 percent of users have completed less than 10 downloads. From Fig. 5.16, it can be obtained that after completing a download, about 85 percent of users seed for an average time of more than 64 hours in the corresponding torrent, and users with the average per torrent seeding time longer than 256 hours compose 55 percent of the total population. Moreover, there are almost 1,000 users with the per torrent seeding time more than 100 days on average, which seems to be unimaginable for public BitTorrent communities.

5.4.2.7 Ratios

We now conduct the analysis on the two types of ratios with respect to user traffic and user time: the share ratio and the ratio of seeding time to leeching time. For simplicity of expression, we represent the latter type of ratio as S/L ratio. Fig. 5.17 depicts the distribution of these two ratios. From this figure, it is obvious to see that most users maintain a satisfactory share ratio: about 88 percent of users upload more



(a) The distribution of accumulative user time (CDF)



(b) The distribution of daily average user time (CDF)

Figure 5.15: The distribution of user time

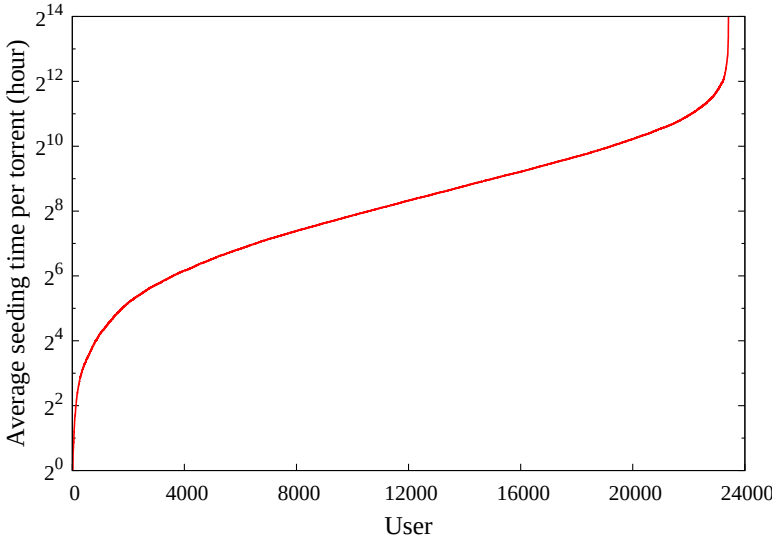


Figure 5.16: Per torrent seeding time for each user

data than they download. Moreover, users with the upload traffic more than twice the download traffic account for more than half of the total population, out of which 20 percent possess a share ratio larger than 8.

Compared with only 12 percent of users with the share ratio less than 1, there are even less (7 percent) users with the S/L ratio less than 1, and in contrast to less than 60 percent of users with the upload traffic more than twice the download traffic, users with S/L ratio larger than 2 form nearly 90 percent of the total population. This staggering contrast between user share ratio and S/L ratio is due to that as shown in Fig. 5.4, in 80 percent of torrents, there are no leechers downloading the corresponding content, and thus by seeding these torrents, users cannot earn upload traffic, whereas the seeding time increases. In addition, as we have shown in Fig. 5.7, the overall ratio of seeders to leechers is roughly 18, which implies each seed has only a little opportunity to serve leechers. Fig. 5.17 also tells that there are about 30 percent of users with a S/L ratio larger than 32, out of which almost half have seeded for 64 times longer than they have leeched, showing a high degree of participation in seeding of these users.

Fig. 5.18 depicts the distribution of snatches and the corresponding upload traffic

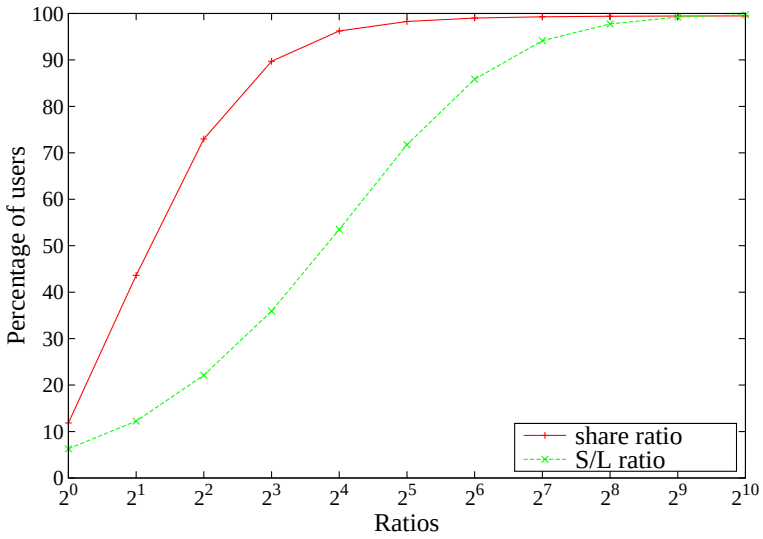


Figure 5.17: The distribution of user share ratio and S/L ratio (CDF)

with respect to per snatch share ratio[1]. As shown in this figure, the 25 percent of snatches with least share ratio almost contribute no upload traffic, and the 73 percent of snatches, with the share ratio less than or equal to 1, account for only less than 20 percent of total upload traffic, which implies that a small part of users may contribute the majority of upload traffic in many torrents. Counter to the intuition that only the high bandwidth users may be able to maintain a share ratio higher than 1 in a single snatch, there are 22,464 users involved in snatches with the share ratio larger than 1, and this is because besides the bandwidth, the upload traffic of a user in a snatch also depends on when this user joined the corresponding torrent and the corresponding leeching and seeding time.

[1]Unlike the user share ratio, the share ratio in each snatch is derived from the upload traffic and the size of the content corresponding to the torrent. In other words, the download traffic in each snatch is counted at no discount

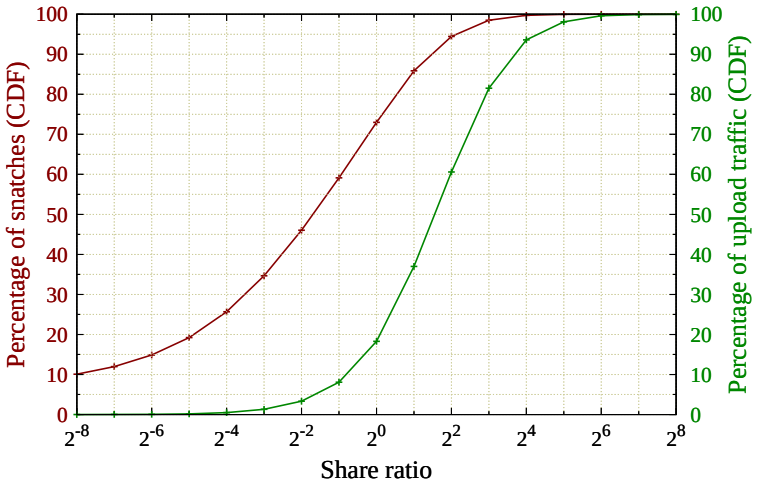


Figure 5.18: Distribution of the snatches and the corresponding upload traffic with respect to share ratio (CDF)

5.4.3 An analysis of user behavior

In this section, we present a detailed analysis on how the bandwidth[1] and age of users influence their behaviors. We conduct this analysis from microscopic and macroscopic perspectives. For microscopic analysis, we intend to find the relationship between the bandwidths of users and the torrents in which they participated; for macroscopic analysis, we want to find the influence of the bandwidth and age of users on their overall seeding and leeching behavior.

5.4.3.1 Microscopic analysis

A torrent is characterized by the content size and popularity, i.e., the number of users completing the download of the corresponding content. We first investigate the relationship between the bandwidths of users and the content sizes of the torrents that they participated in. To simplify the analysis, we classify users and torrent into 8 and

[1]Since in each snatch, the information of the corresponding user and torrent, and the corresponding upload and download rate are available, we obtain for each user the download and upload rate in all snatches in which this user is involved, and use the maximum value of all these measured rates as the bandwidth of this user.

10 classes, respectively. In particular, a user with a bandwidth of n Kbytes/s belongs to class $\min\{\max\{0, \lceil \log_2 \frac{n}{256} \rceil\}, 7\}$, and a torrent with a content size of s GB belongs to class $\min\{\max\{0, \lceil \log_2 s \rceil\}, 9\}$.

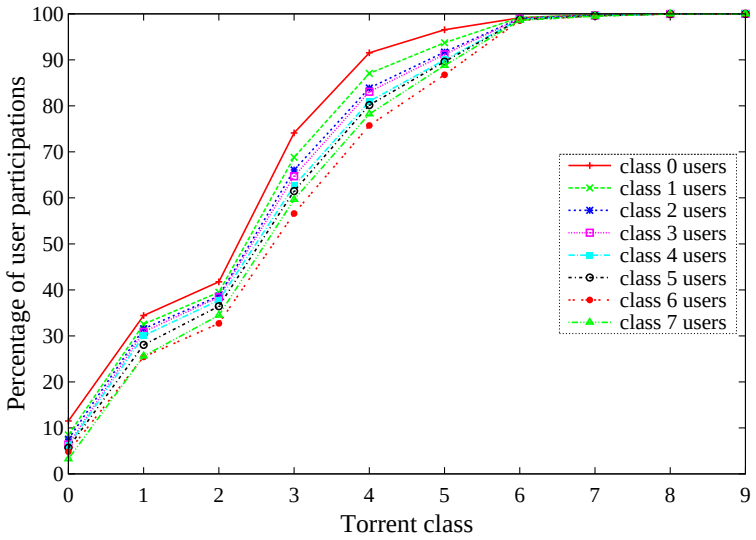


Figure 5.19: The distribution of user participations with respect to torrent size (CDF)

Fig. 5.19 demonstrates the distribution of user participations[1] with respect to torrent size. It is clearly shown in this figure that *lower class (bandwidth) users participated in the torrents with smaller content sizes more often than higher class (bandwidth) users*. This is probably because downloading the torrent with a large content size is less cost-effective for lower bandwidth users than for higher bandwidth users. More specifically, it is difficult to earn upload traffic for low bandwidth users, and moreover, downloading the torrent with a large content size will result in a significant increase in the download traffic, which in turn dramatically decreases the share ratio.

We then explore the distribution of users of different classes in torrents with different popularities. To this end, we first re-classify torrents into 41 classes in the way that a torrent with the number of user participations being n belongs to class $\min\{\lceil \frac{n}{100} \rceil - 1, 40\}$, and then plot the distribution of user participations with respect to torrent

[1] A user participation corresponds to a snatch.

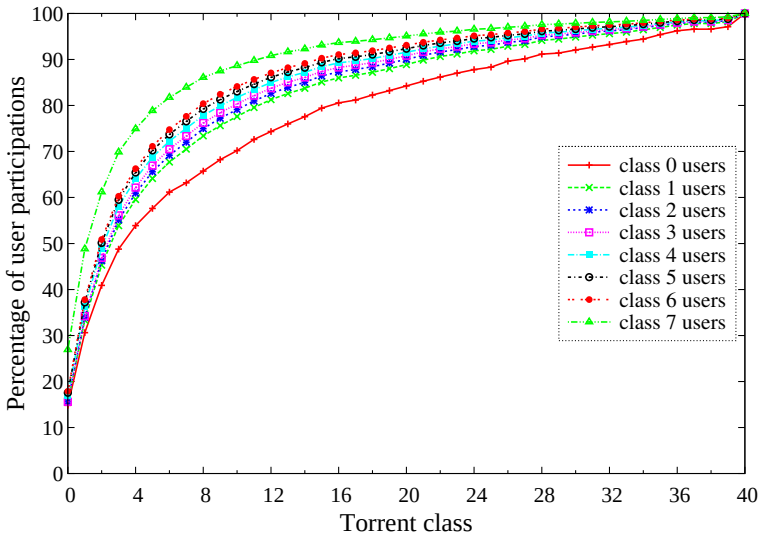


Figure 5.20: The distribution of user participations with respect to torrent popularity (CDF)

class in Fig. 5.20. It can be observed from this figure that *lower bandwidth users are more likely to participate in high-popularity torrents than higher bandwidth users*. This is easy to understand. For high bandwidth users, it is not difficult to earn upload traffic even there are only a few peers in the torrent, and hence they are less sensitive to torrent popularity in choosing which torrents to participate in. However, for low bandwidth users, it is much easier to earn upload traffic in high-popularity torrents than in low-popularity ones, as there are much more peers in high-popularity torrents.

5.4.3.2 Macroscopic analysis

We now investigate how the overall seeding and leeching behavior of users are influenced by users' bandwidth and age. To this end, we classify the users according to the age and bandwidth in the way that a user with an age of m weeks and with a measured bandwidth of n Kbytes/s belongs to class (i, j), where i and j are defined as follows:

$$i = \max\{0, \lceil \log_2 m \rceil\}$$

$$j = \min\{\max\{0, \lceil \log_2 \frac{n}{256} \rceil\}, 7\}$$

In addition, We stipulate that a class (i_1, j_1) is higher than class (i_2, j_2) if one of the following two conditions is satisfied.

$$i_1 > i_2, \text{ and } j_1 \geq j_2,$$
$$i_1 \geq i_2, \text{ and } j_1 > j_2,$$

The distribution of users among different classes is shown in Fig. 5.21. It can be shown that there are at least 2 users in each class, and only 7 classes have a number of users less than 10. Thus for most classes, there are enough samples for us to study the overall behaviors of users in the same class.

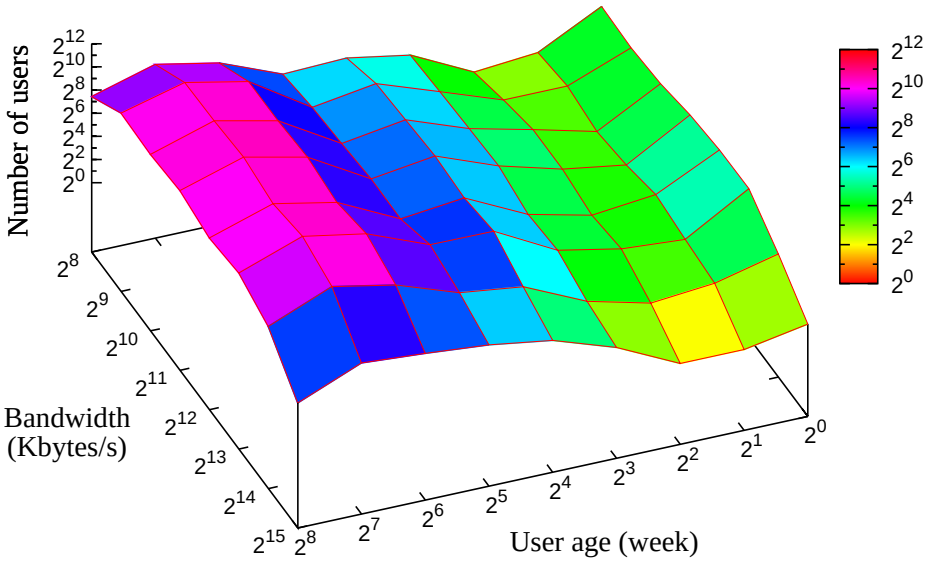


Figure 5.21: The distribution of user among different classes

We first study how user age and bandwidth exert influence on the number of completed downloads. We average the numbers of completed downloads of users in each class, and show the result in Fig. 5.22. It can be seen from this figure that both the

age and bandwidth of a user may affect this user's download behavior. While it is natural that an older user tends to participate in more torrents than a younger one, it may seem to be a little out of expectation that *a user is likely to have completed less downloads of torrents than users with a higher bandwidth have done.* As can be observed in Fig. 5.22, for users belonging to the class (i, j), where $i \geq 3$, the average number of completed downloads is larger than that of users in the class (i, k), where $k < j$. This can be attributed to that the high bandwidth users may join multiple torrents simultaneously in order to get their bandwidths effectively utilized.

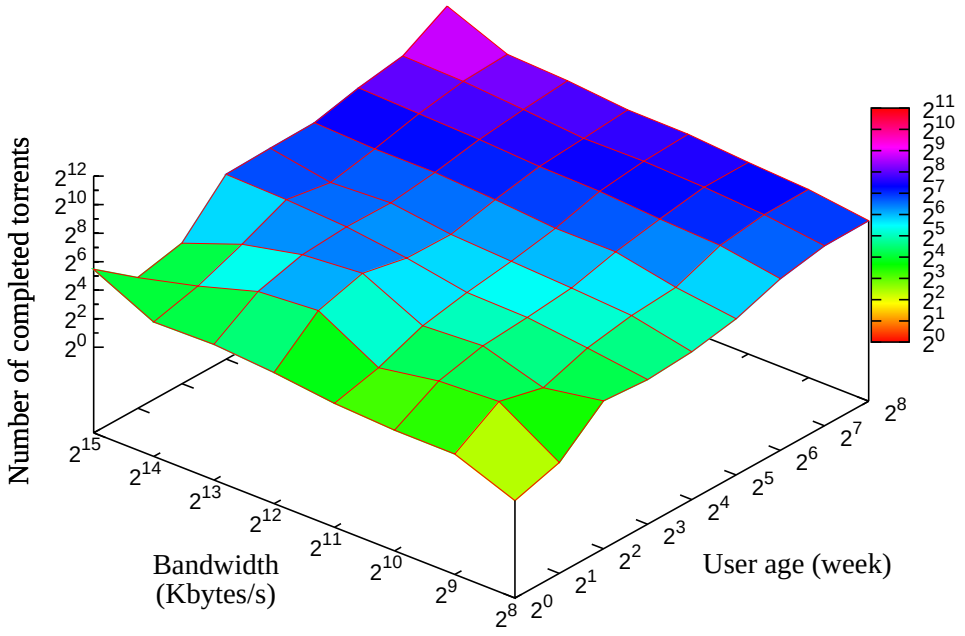


Figure 5.22: Number of completed downloads versus user age and bandwidth

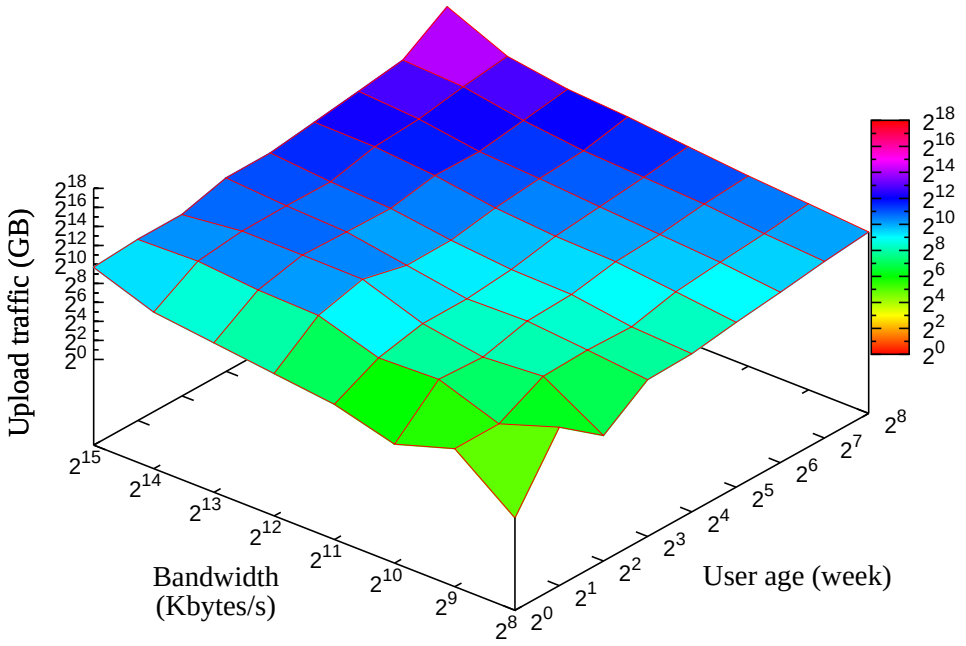
The average number of completed downloads of some classes may be larger than that of higher classes. E.g., users in class $(2, 4)$ tend to have completed more downloads than users in class $(2, 5)$ or $(3, 4)$ have done. This may have multiple reasons. First, the users in these classes have joined CHDBits for a short period, and thus the average number of completed downloads of these classes may not be able to reflect users'

long-term download behavior. Second, as shown in Fig. 5.21, there are only a few users in some classes, which may result in a deviation in the average number of completed downloads in these classes. Third, as we just stated, users may simultaneously participate in multiple torrents, and as a consequence, the bandwidth of these users is shared by multiple tasks, which leads to the measured bandwidth being less than the real value. Finally, there may exist some very active users in a particular class, which have participated in much more torrents than users in higher classes have done.

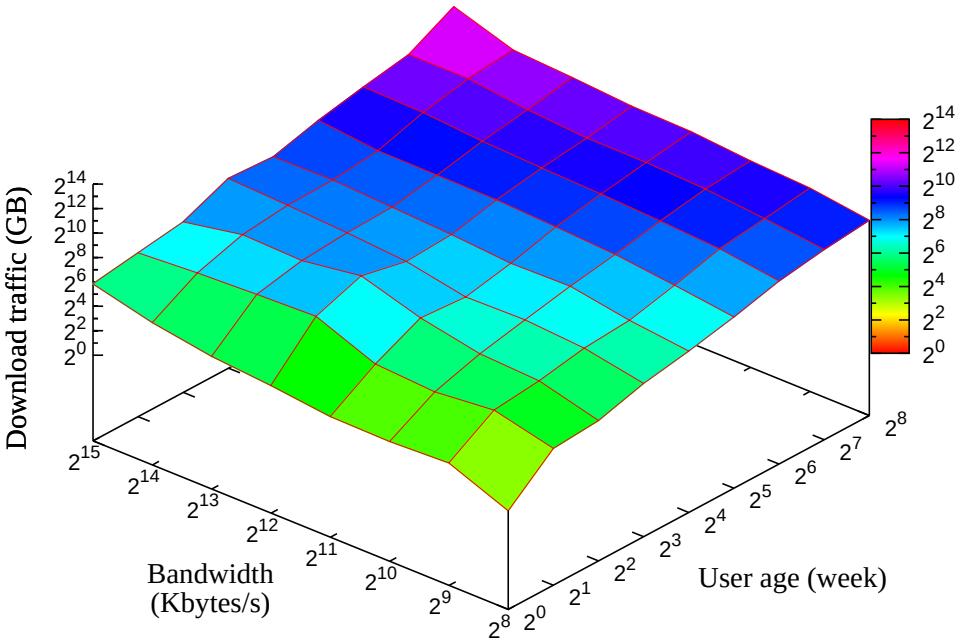
We then conduct an investigation on the influence of the age and bandwidth of a user on its traffic. Fig. 5.23a and 5.23b demonstrate the average upload and download traffic of users in the same class, respectively, and it can be shown that both figures highly coincide with Fig. 5.22. More specifically, the average user upload and download traffic correspond to the average number of completed downloads of users in the same class, and vice versa.

It can also be shown that Fig. 5.23b dovetails more with Fig. 5.22 than Fig. 5.23a does. As can be observed from these figures, the age of users exerts more influence on download traffic and number of completed downloads of users than user bandwidth does, while with respect to the upload traffic, both the age and bandwidth of a user have similar impact. This is also easy to understand. The download traffic of a user almost completely depends on the torrents it participated in. However, the upload traffic a user contributes in a torrent also depends on its bandwidth, and generally, a user with a higher bandwidth earns more upload traffic than a user with lower bandwidth does in the same torrent. Consequently, the upload traffic of a high bandwidth user may be similar to, or even more than that of a low bandwidth user who has participated in much more torrents than the high bandwidth user has done.

There also exists inconsistence between Fig. 5.22 and Fig. 5.23a. For example, while the number of completed downloads of users in class $(0, 1)$ is roughly the same as that in class $(0, 2)$, the average user upload traffic of class $(0, 1)$ is abnormally much



(a) User upload traffic versus user age and bandwidth

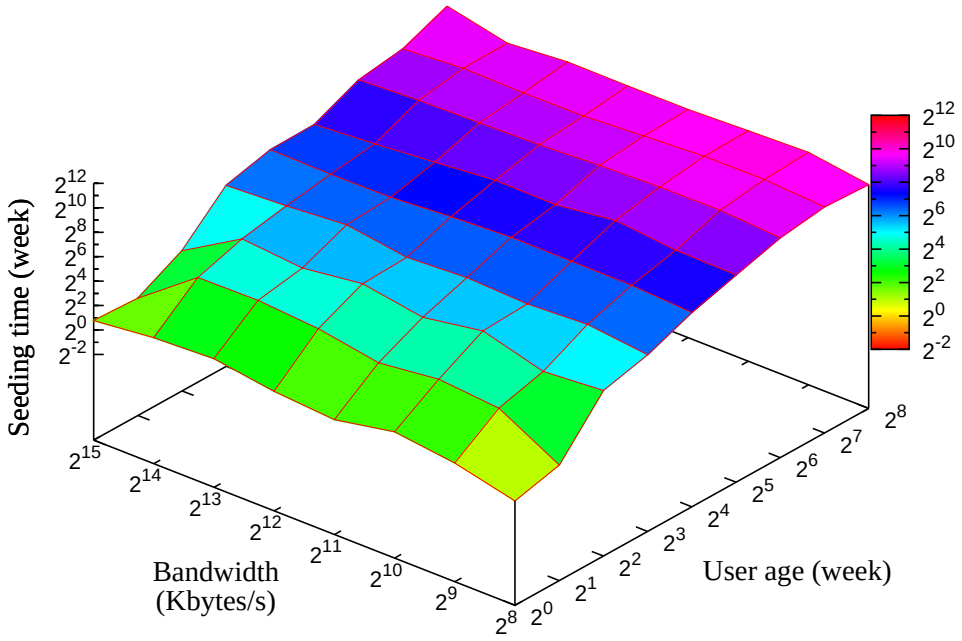


(b) User download traffic versus user age and bandwidth

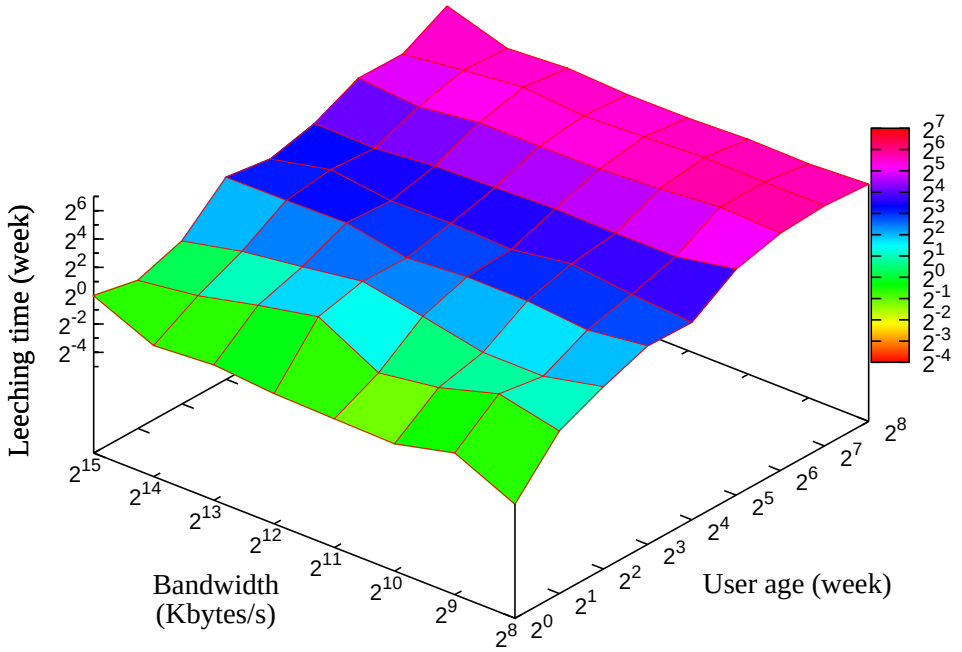
Figure 5.23: The influence of user age and bandwidth on upload and download traffic

higher than that in class $(0, 2)$; a similar phenomenon also exists between class $(1, 0)$ and $(1, 1)$. This may also be attributed to user's simultaneous downloading of multiple torrents, or the existence of very active users in some particular user classes. Another explanation to this abnormal phenomenon is that some users may cheat the tracker by reporting the modified traffic. In order to confirm this conjecture, we examine the user data of snapshot 2011/09/03, and find that a class (0,1) user with the measured bandwidth of 451 Kbytes/s and the total user time of 275 hours claims that it has uploaded more than 800 GB of data. Moreover, the data collection carried out on 2011/09/09 reveals that while the total time this user spends in seeding and leeching doubled in the six-day period, the upload traffic was increased by only 12 GB.

The effects of user age and bandwidth on user time are also described in Fig. 5.24. As a result of the participation in more torrents, older users are likely to spend more time on seeding and leeching than younger users do. However, contrary to what it does to user traffic and average number of completed downloads, *the bandwidth of user exerts a negative influence on user's seeding time.* As shown in Fig. 5.24, for $i \geq 3$, the seeding time of class (i, j) users is less than that of users in class (i, k), given $k < j$, although users in the former class tend to participate in more torrents than users in the latter class. This can probably be attributed to the policy that CHDBits users are allowed to trade the points, which can be earned by seeding, for upload traffic. This policy may have little attraction for high bandwidth users for two reasons. First, the high bandwidth users can earn a great amount of upload traffic in a short time when there are enough leechers present, since at this time their outgoing bandwidth can be effectively utilized. Second, when most leechers complete the download, continuing to seed would not result in a proportional increase in upload traffic, which can also be inferred from Fig. 5.4. However, the low bandwidth users may seem to be motivated by this policy, since it may be difficult for them to get enough upload traffic via seeding and leeching, and this policy provides another way to increase the upload traffic.



(a) User seeding time versus user age and bandwidth



(b) User leeching time versus user age and bandwidth

Figure 5.24: The influence of user age and bandwidth on seeding and leeching time

In addition, for $i \geq 5$, the high bandwidth users in class (i, j) may spend less time on leeching than users in class (i, k), where $k < j$, and this is easy to understand since high bandwidth users could download at a higher rate than low bandwidth users do.

5.5 Conclusion

Private BitTorrent community is becoming increasingly popular, and generates a huge amount of Internet traffic. In this chapter, we present a thorough survey on a famous private BitTorrent community, CHDBits. We find that the content of most torrents are available, although many torrents may have been released for a very long time, and we also obtain that users in most torrents could download at a high rate. By studying user profiles, we show that many users have participated in multiple torrents, thereby generating a great amount of upload and download traffic. The user profiles also reveal that almost all users maintain a high share ratio, and are willing to contribute to content distribution by spending more time in seeding than in leeching. Our survey suggests that the bandwidth of a user have a significant impact on user behavior. Low bandwidth users are more likely to participate in the torrents with a smaller content size or a higher popularity; compared with low bandwidth users, high bandwidth users tend to participate in more torrents, but spend less time in seeding.

Chapter 6

Conclusion and Future Work

In this thesis, we have carried out a study on the analysis and improvement of the performance of BitTorrent swarms, with a special concentration on content availability. In Chapter 3, we quantify the performance metrics related to content availability of BitTorrent swarms and the implications of content bundling. The analysis reflects that bundling is a promising way to enhance the content availability and can even reduce the sojourn time of peers under some specific scenarios where content is highly unavailable. We then perform a study on the feasibility of using network coding to ameliorate the content availability of BitTorrent swarms in Chapter 4. This study shows that network coding has significant potential to enhance the content availability of BitTorrent swarms, but the existing coding schemes are not feasible due to some fundamental deficiencies. To overcome these deficiencies, we present a sparse network coding scheme and a block scheduling algorithm, and verify the efficiency of the proposed coding scheme and algorithm by performing extensive simulations. In Chapter 5, we present a detailed survey on a large private BitTorrent community, which is becoming increasingly popular recently. By examining multiple different aspects of torrents and users, our survey demonstrates the promising performance of swarms in this community, develops an in-depth understanding to how users participating in downloading and uploading, and reveals some new findings with respect to user behaviors.

6.1 Future work

6.1.1 Bundled swarm vs. individual swarm

Although there has been study [42] empirically comparing the performance of bundled swarms and that of individual swarms, the comparison, however, is not conducted in a fair way, as there is no correlation between the content distributed in bundled swarms and that in individual swarms. Therefore, how BitTorrent users' selection of files to be downloaded is affected in bundled swarms has not yet been reflected, and the improvement in content availability caused by content bundling in real-world scenarios is also open.

6.1.2 Content propagation

A great amount of content, originated from BitTorrent swarms, has become available in Internet [31]. It is interesting to study how this content propagates to Internet. In addition, this propagation improves the availability of content, and another interesting question is thus to quantify this improvement and to leverage this improved availability of content in Internet to enhance the download performance of BitTorrent peers.

6.1.3 Strategic manipulation of upload slots

The number of upload slots of many BitTorrent peers is set to four by default; however, it is unlikely that four upload slots could saturate the outgoing bandwidth of peers, especially in the heterogeneous BitTorrent swarms. Therefore, for many peers, there may exist some bandwidth resource that has not been effectively utilized. By allocating this idle bandwidth resource to neighbours other than those already being unchoked, a peer can increase the probability of being unchoked by these extra unchoked neighbors in the following rounds, thereby enhancing the download performance. As a result, increasing the number of upload slots leads to a double win situation since the overall swarming efficiency is also improved due to more bandwidth resource. Therefore, it is

of great help to explore the manipulation space of upload slots for peers with different bandwidth resource, and see if there exists a strategy which leads to Nash equilibrium where optimal swarming efficiency is achieved.

6.1.4 Exploration on other feasible linear network coding schemes

Although we have already presented a feasible linear network coding scheme which can significantly improve the content availability of BitTorrent swarms while incurring linear computation and disk read/write cost, there might exist some other coding schemes that are also feasible from the perspectives of the improvement and cost incurred. We intend to explore and design different sparse coding schemes which incur acceptable computation and disk read/write costs, and then perform mathematical analysis and extensive experiments to investigate their impact on the content availability of BitTorrent swarms. For the feasible coding schemes, we will characterize their common features, and use these features to guide the design of feasible coding schemes.

Appendix A

Modeling and Analysis of Content Availability and Bundling in BitTorrent-like File Swarming Systems

A.1 Background

We base our analysis on the result of the C-congestion period of an M/M/∞ queue reported in [86], in which a C-congestion period is defined as "the period starting at the epoch that an arriving customer finds C customers in the system, until the first time that a departing customer leaves behind C customers."

In an M/M/∞ queue with the arrival rate and mean service time being λ and $\frac{1}{\mu}$, respectively, let Λ_t denote the number of customers in the queue at time t, and let

$$D_j(i) := \inf\{\Delta t : \Lambda_{t_0+\Delta t} = j | \Lambda_{t_0} = i\}, i > j \tag{A.1}$$

Then, as shown in [86],

$$D_j(i) = \sum_{k=j}^{i-1} D_k(k+1) \tag{A.2}$$

where $D_k(k+1)$ is exactly a k-congestion period. According to [86] and the result of residual busy period in [70], we further have

$$E[D_k(k+1)] = \frac{k!}{\lambda \rho^k} \sum_{j=k+1}^{\infty} \frac{\rho^j}{j!} \tag{A.3}$$

where $\rho = \frac{\lambda}{\mu}$.

A.2 Proof

A.2.1 Proof of Lemma 3.1

Proof. Given that $\frac{1}{\gamma} \to \infty$, and peers immediately depart the swarm once they finish the download, if there are i, $i \geq r$, peers left when the publisher departs the swarm, then by its definition, $D_{r-1}(i)$ is exactly the duration of residual active period after the leave of the publisher, and the mean of $D_{r-1}(i)$ can be derived by using equation (A.2) and (A.3) as follows:

$$E[D_{r-1}(i)] = \sum_{j=r-1}^{i-1} \frac{j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} \tag{A.4}$$

When the publisher leaves, the number of online peers is Poisson distributed with mean of $\frac{\lambda s}{u}$ since we have assumed peer population is in steady state at that time. Thus, given $\frac{1}{\gamma} \to \infty$, the mean length of the residual active period after the leave of the publisher in the individual swarm with selfish peers, $E[R]$, is

$$\begin{aligned} E[R] &= \sum_{i=r}^{\infty} \frac{(\frac{\lambda s}{u})^i}{i!} e^{-\frac{\lambda s}{u}} E[D_{r-1}(i)] \\ &= \sum_{i=r}^{\infty} \frac{(\frac{\lambda s}{u})^i e^{-\frac{\lambda s}{u}}}{i!} \sum_{j=r-1}^{i-1} \frac{j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} \end{aligned} \tag{A.5}$$

The proof of equation (3.2) is same as above. ∎

A.2.2 Proof of Theorem 3.2

Proof. An active period starts when the publisher enters the swarm terminating the current passive period. Once the publisher departs the swarm, as we have assumed that the length of residual active period fluctuates slightly around its mean, the publisher will reenter the swarm before peer population falls below r with a probability about $1 - e^{-\gamma E[R]}$. In this case, another active period is initiated with the same mean as the

original one. When the number of peers drops below r, if the publisher has not arrived, the active period is then terminated.

Let H denote the event that the publisher reenters the swarm before peer population falls below r, and let $\bar{H}$ denote the complementary event of H. H occurs with a probability of $1 - e^{-\gamma E[R]}$, and $\bar{H}$ occurs with a probability of $e^{-\gamma E[R]}$. Let t denote the length of publisher idle period. The mean length of active periods, $E[A]$, thus can be given by

$$E[A] = \frac{1}{\beta} + P(H)(E[t|H] + E[A]) + P(\bar{H})E[R] \tag{A.6}$$

The conditional probability density of t is

$$f(t|H) = \begin{cases} \frac{\gamma e^{-\gamma t}}{1-e^{-\gamma E[R]}}, & \text{if } t \leq E[R] \\ 0, & \text{if } t > E[R] \end{cases} \tag{A.7}$$

Then the conditional mean of t is

$$\begin{aligned} E[t|H] &= \int_0^{E[R]} \frac{\gamma e^{-\gamma t}}{1 - e^{-\gamma E[R]}} t \, \mathrm{d}t \\ &= \frac{1}{\gamma} - \frac{e^{-\gamma E[R]}}{1 - e^{-\gamma E[R]}} E[R] \end{aligned} \tag{A.8}$$

Substituting equation (A.8) into (A.6) yields

$$E[A] = \frac{1}{\beta} + (1 - e^{-\gamma E[R]})(\frac{1}{\gamma} - \frac{e^{-\gamma E[R]}}{1 - e^{-\gamma E[R]}} E[R] + E[A]) + e^{-\gamma E[R]} E[R] \tag{A.9}$$

Equation (3.3) can be obtained by solving this equation.

Equation (3.4) can be proved in the same way. ∎

A.2.3 Proof of Theorem 3.3

Proof. Let

$$m_j = \frac{j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} \tag{A.10}$$

$$M_j = \frac{j!}{N\lambda(\frac{N^2\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{N^2\lambda s}{u})^k}{k!} \tag{A.11}$$

It is easy to see that

$$\begin{aligned} M_j &= \frac{j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} N^{2(k-j)-1} \\ &> \frac{N \times j!}{\lambda(\frac{\lambda s}{u})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda s}{u})^k}{k!} = N m_j \end{aligned} \tag{A.12}$$

and equation (3.1) and (3.2) ca be rewritten as

$$E[R] = \sum_{i=r}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} \sum_{j=r-1}^{i-1} m_j = \sum_{j=r-1}^{\infty} m_j \sum_{i=j+1}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} \tag{A.13}$$

$$E[\mathcal{R}] = \sum_{i=r}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} \sum_{j=r-1}^{i-1} M_j = \sum_{j=r-1}^{\infty} M_j \sum_{i=j+1}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} \tag{A.14}$$

In order to show $\frac{E[\mathcal{R}]}{E[R]} > N$, it suffices to prove $\sum_{i=j+1}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} < \sum_{i=j+1}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!}$.

Since

$$\sum_{i=0}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} = \sum_{i=0}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} = 1 \tag{A.15}$$

we only need to prove $\sum_{i=0}^{j} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} > \sum_{i=0}^{j} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!}$. To this end, we first assume $\sum_{i=0}^{j} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} \leq \sum_{i=0}^{j} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!}$.

As i increases, $e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i$ grows faster than $e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i$. Thus, for all $i > \frac{e^{(N^2-1)\lambda s}}{2u\ln N}$,

$$e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i > e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i$$

and for all $i < \frac{e^{(N^2-1)\lambda s}}{2u\ln N}$,

$$e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i < e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i$$

Since $\sum_{i=0}^{j} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} \leq \sum_{i=0}^{j} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!}$, and $N > 1$, we have

$$e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^j > e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^j$$

Thus, $j > \frac{e^{(N^2-1)\lambda s}}{2u\ln N}$, and for all $i > j$, $e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i > e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i$. Then we have

$$\sum_{i=0}^{j} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} + \sum_{i=j+1}^{\infty} \frac{e^{-\frac{\lambda s}{u}}(\frac{\lambda s}{u})^i}{i!} < \sum_{i=0}^{j} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} + \sum_{i=j+1}^{\infty} \frac{e^{-\frac{N^2\lambda s}{u}}(\frac{N^2\lambda s}{u})^i}{i!} \quad \text{(A.16)}$$

which contradicts equation (A.15). ∎

A.2.4 Proof of Theorem 3.4

Proof. By its definition, the content availability in the individual swarm, P, is

$$P = \frac{E[A]}{E[A] + \frac{1}{\gamma}} = 1 - \frac{1/\gamma}{(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]}} = 1 - \frac{1}{\gamma(\frac{1}{\beta} + \frac{1}{\gamma})e^{\gamma E[R]}}$$

Equation (3.7) can be proved in the same way. ∎

A.2.5 Proof of Theorem 3.5

Proof. As stated before, the sojourn time of a peer primarily depends on the download time and the waiting time before it can start download. Since peer arrival process is a Poisson process, according to PASTA property (Poisson Arrivals See Time Averages), the proportion of peers accounted for by those arriving in passive periods is exactly the same as the proportion of the length of passive periods to the swarm's lifetime, which by its definition is content unavailability. Therefore, the average sojourn time of peers in the individual swarm is given by

$$E[T] = T_{\text{download}} + T_{\text{waiting}} = \frac{s}{u} + \frac{1}{\gamma}(1 - P) = \frac{s}{u} + \frac{\beta/\gamma}{(\beta + \gamma)e^{\gamma E[R]}}$$

Equation (3.9) can be proved in the same way. ∎

A.2.6 Proof of Theorem 3.6

Proof. If $2 \leq N \leq \frac{\beta/\gamma}{(\beta+\gamma)e^{\gamma E[R]}}\left(1-e^{-\gamma E[R]}\right)\frac{u}{s}+1$, then

$$\begin{aligned} E[\mathcal{T}]-E[T] &= \frac{(N-1)s}{u}-\frac{\beta/\gamma}{\beta+\gamma}\left(e^{-\gamma E[R]}-e^{-\gamma E[\mathcal{R}]}\right) \\ &< \frac{(N-1)s}{u}-\frac{\beta/\gamma}{\beta+\gamma}\left(e^{-\gamma E[R]}-e^{-\gamma NE[R]}\right) \\ &\leq \frac{(N-1)s}{u}-\frac{\beta/\gamma}{\beta+\gamma}\left(e^{-\gamma E[R]}-e^{-2\gamma E[R]}\right) \\ &\leq 0 \end{aligned}$$

A.2.7 Proof of Lemma 3.7

Proof. Given that $\frac{1}{\gamma} \to \infty$, when the publisher leaves the swarm, the residual active period can be divided into several sub-periods. The period between the leave of the publisher and the epoch that the last seed in the second queue leaves is the first sub-period. If there are k seeds in the second queue at the departure of the publisher, then the length of the first sub-period is $D_0(k)$. As we have assume that the Jackson network is in steady state when the publisher leaves, we can thus apply the method described in the proof of Lemma 3.1 in solving the mean length of the first sub-period. By setting the value of r to 1 and replacing $\frac{s}{u}$ with $\frac{1}{\theta}$ in equation (3.1), we get the expression of the mean length of the first sub-period

$$E[R_{first}] = \sum_{i=1}^{\infty}\frac{e^{-\lambda/\theta}(\frac{\lambda}{\theta})^i}{i!}\sum_{j=0}^{i}\frac{j!}{\lambda(\frac{\lambda}{\theta})^j}\sum_{k=j+1}^{\infty}\frac{(\frac{\lambda}{\theta})^k}{k!} \tag{A.17}$$

The second sub-period follows the first one if there are no less than r leechers in the first queue at the end of the first sub-period. In the second sub-period, a leech in the first queue will sooner or later complete the download and then enter the second queue instantaneously. Assuming the first queue is in steady state at the beginning

of the second sub-period[1], the input of the second queue is still a Poisson process, and the peer finishing the download earliest among all leechers in the first queue will enter the second queue and initiate a busy period; the second sub-period ends with the termination of this busy period. Therefore, the second sub-period consists of the waiting time before the first peer enters the second queue, which is an exponentially distributed variable with the mean of $\frac{1}{\lambda}$, and a busy period initiated by the arrival of this peer. Thus the mean length of the second sub-period is

$$E[R_{second}] = \frac{1}{\lambda} + \frac{e^{\lambda/\theta} - 1}{\lambda} = \frac{e^{\lambda/\theta}}{\lambda} \tag{A.18}$$

At the end of the second sub-period, if the last departing seed in the second queue once again finds no less than r leechers existing in the first queue, another sub-period, which is identical to the current one, i.e., the second sub-period, will be initiated. As a result, given that there are no less than r peers in the first queue at the end of the first sub-period, the number of occurrence of the second sub-period is geometrically distributed. In addition, at the end of each sub-period, as the Jackson network is in steady state, the probability that peer population falls below r, denoted by P_r, is $\sum_{i=0}^{r-1} \frac{(\lambda s/u)^i}{i!} e^{-\lambda s/u}$.

Given above statements, we have

$$E[R] = E[R_{first}] + \frac{1 - P_r}{P_r} E[R_{second}] \tag{A.19}$$

Substituting equation (A.17) and (A.18) into (A.19) yields (3.11). Equation (3.12) can be proved in the same way. ∎

[1]In fact, at the end of the first sub-period, the first queue is in steady state, but the second sub-period is initiated only if there are at least r peers in the first queue at this time. Thus, at the beginning of the second sub-period, the probability that there are k, $k \geq r$, peers in the first queue is $\frac{(\lambda s/u)^k/k!}{\sum_{i=r}^{\infty}(\lambda s/u)^i/i!}$, which indicates that the first queue is not in steady state. However, as long as the mean of the length of the first queue is much greater than population threshold, r, the possibility that peer population falls below r in steady state is negligible, and thus the distribution of peer population at the beginning of the second sub-period is nearly the same as the distribution in steady state.

A.2.8 Proof of theorem 3.8

Proof. For short-hand notation, we let

$$R_1 = \sum_{i=1}^{\infty} \frac{e^{-\lambda/\theta}(\frac{\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \frac{j!}{\lambda(\frac{\lambda}{\theta})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{\lambda}{\theta})^k}{k!} \tag{A.20}$$

$$R_2 = \frac{1-P_r}{P_r} \frac{e^{\lambda/\theta}}{\lambda} \tag{A.21}$$

$$\mathcal{R}_1 = \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \frac{j!}{N\lambda(\frac{N\lambda}{\theta})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{N\lambda}{\theta})^k}{k!} \tag{A.22}$$

$$\mathcal{R}_2 = \frac{1-\mathcal{P}_r}{\mathcal{P}_r} \frac{e^{N\lambda/\theta}}{N\lambda} \tag{A.23}$$

where $P_r = e^{-\lambda s/u} \sum_{i=0}^{r-1} \frac{(\lambda s/u)^i}{i!}$, and $\mathcal{P}_r = e^{-N^2\lambda s/u} \sum_{i=0}^{r-1} \frac{(N^2\lambda s/u)^i}{i!}$. Then equation (3.11) and (3.12) can be rewritten as

$$E[R] = R_1 + R_2 \tag{A.24}$$

$$E[\mathcal{R}] = \mathcal{R}_1 + \mathcal{R}_2 \tag{A.25}$$

As R_1 and R_2 are functionally independent of N, we have $E[R] = \Theta(1)$. We then discuss the bounds of $\mathcal{R}_1$ and $\mathcal{R}_2$. For $\mathcal{R}_1$, we have

$$\begin{aligned}
\mathcal{R}_1 &= \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \frac{j!}{N\lambda(\frac{N\lambda}{\theta})^j} \sum_{k=j+1}^{\infty} \frac{(\frac{N\lambda}{\theta})^k}{k!} \\
&= \frac{1}{N\lambda} \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \sum_{k=1}^{\infty} \frac{(\frac{N\lambda}{\theta})^k}{\frac{(k+j)!}{j!}} \\
&< \frac{1}{N\lambda} \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} \sum_{j=0}^{i} \sum_{k=1}^{\infty} \frac{(\frac{N\lambda}{\theta})^k}{k!} \\
&< \frac{1}{N\lambda} \sum_{i=1}^{\infty} \frac{e^{-N\lambda/\theta}(\frac{N\lambda}{\theta})^i}{i!} (i+1) e^{N\lambda/\theta} \\
&= \frac{1}{N\lambda} \sum_{i=1}^{\infty} \frac{(i+1)(N\lambda/\theta)^i}{i!} < (\frac{1}{N\lambda} + \frac{1}{\theta}) e^{N\lambda/\theta}
\end{aligned}$$

and since

$$\begin{aligned}\lim_{N\to\infty} N\lambda \times \mathcal{P}_r &= \lim_{N\to\infty} N\lambda \times e^{-N^2\lambda s/u} \sum_{i=0}^{r-1} \frac{(N^2\lambda s/u)^i}{i!} \\ &< \lim_{N\to\infty} N\lambda \times e^{-N^2\lambda s/u} \sum_{i=0}^{r-1} (N^2\lambda s/u)^i \\ &< \lim_{N\to\infty} (N^2\lambda s/u)^r e^{-N^2\lambda s/u}\end{aligned}$$

we have

$$\lim_{N\to\infty} \mathcal{R}_2 = \lim_{N\to\infty} \frac{1-\mathcal{P}_r}{\mathcal{P}_r}\frac{e^{N\lambda/\theta}}{N\lambda} = \lim_{N\to\infty} \frac{1}{N\lambda\mathcal{P}_r} e^{N\lambda/\theta} > \lim_{N\to\infty} \frac{e^{N\lambda/\theta+N^2\lambda s/u}}{(N^2\lambda s/u)^r}$$

Therefore,

$$\lim_{N\to\infty} \log E[\mathcal{R}] > \lim_{N\to\infty} \log \mathcal{R}_2 > \lim_{N\to\infty} \log \frac{e^{N\lambda/\theta+N^2\lambda s/u}}{(N^2\lambda s/u)^r} = \Theta(N^2) \tag{A.26}$$

On the other hand, since $r > 1$,

$$\mathcal{R}_2 = \frac{1-\mathcal{P}_r}{\mathcal{P}_r}\frac{e^{N\lambda/\theta}}{N\lambda} < \frac{1}{e^{-N^2\lambda s/u}}\frac{e^{N\lambda/\theta}}{N\lambda} = \frac{e^{N\lambda/\theta+N^2\lambda s/u}}{N\lambda}$$

and thus

$$\log E[\mathcal{R}] < \log\left((\frac{1}{N\lambda}+\frac{1}{\theta})e^{N\lambda/\theta} + \frac{e^{N\lambda/\theta+N^2\lambda s/u}}{N\lambda}\right) = \Theta(N^2) \tag{A.27}$$

Since $E[R] = \Theta(1)$, given (A.26) and (A.27), we are able to conclude that

$$\log \frac{E[\mathcal{R}]}{E[R]} = \Theta(N^2)$$

∎

APPENDIX A

Appendix B

Mathematical Analysis on the Effect of Network Coding on the Performance of BitTorrent Swarms

B.1 Proof of Lemma 4.1

Proof. It is easy to see that when $i = 0$, $\Pr(\mathbb{A}_{i,j}) = 1$, and $\Pr(\mathbb{A}_{i,j}) = 0$ if $i > j$. To validate the reminder of (4.2), consider the sets of coded blocks at two peers: $b = \{b_1, b_2, \ldots, b_i\}$, and $b' = \{b'_1, b'_2, \ldots, b'_j\}$. Each element in set b is a linear combination of elements in set b', which also implies $i \leq j$. Now suppose the first peer has downloaded a new coded block b_{i+1}. Since i blocks can generate q^i linear combinations, the new block b_{i+1} is restricted to be selected from $q^n - q^i$ other blocks, out of which there are $q^j - q^i$ blocks that are linear combinations of elements of set b'. Therefore, the probability of $\mathbb{A}_{i+1,j}$ conditioned on $\mathbb{A}_{i,j}$ is

$$\Pr(\mathbb{A}_{i+1,j}|\mathbb{A}_{i,j}) = \frac{q^j - q^i}{q^n - q^i} \tag{B.1}$$

and thus $\Pr(\mathbb{A}_{i,j})$ can be derived as follows:

$$\begin{aligned}
&\Pr(\mathbb{A}_{i,j}) \\
&= \Pr(\mathbb{A}_{i,j}|\mathbb{A}_{i-1,j})\Pr(\mathbb{A}_{i-1,j}) \\
&= \Pr(\mathbb{A}_{i,j}|\mathbb{A}_{i-1,j})\Pr(\mathbb{A}_{i-1,j}|\mathbb{A}_{i-2,j}) \ldots \Pr(\mathbb{A}_{1,j}|\mathbb{A}_{0,j})\Pr(\mathbb{A}_{0,j}) \\
&= \frac{q^j - q^{i-1}}{q^n - q^{i-1}} \frac{q^j - q^{i-2}}{q^n - q^{i-2}} \cdots \frac{q^j - 1}{q^n - 1}
\end{aligned}$$

∎

B.2 Proof of Theorem 4.2

Proof.

$$
\begin{aligned}
&\Pr(\textit{a peer is usable to another peer}) \\
&= \sum_{i=0}^{n} \Pr(p=i) \sum_{j=0}^{n-1} \Pr(p=j)(1-\Pr(\mathbb{A}_{i,j})) \\
&= \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{n} (1-\Pr(\mathbb{A}_{i,j})) \\
&= \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{j} \Pr(\mathbb{A}_{i,j}) \qquad \text{(B.2)} \\
&> \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{j} \frac{1}{q^{(n-j)i}} \\
&> \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} (1+\frac{1}{q^{n-j}}) \\
&> \frac{n(n+1)}{(n+1)^2} - \frac{n+1/q}{(n+1)^2} > \frac{n^2-1}{(n+1)^2} = \frac{n-1}{n+1}
\end{aligned}
$$

∎

B.3 Proof of Lemma 4.3

Proof. It is not difficult to solve the probability of $\mathbb{A}_{i,j}$ when $i = 0$ or $j < i \leq n$. We therefore focus on the situation when $0 < i \leq j$. When $i = j = 1$, it is easy to show that $\Pr(\mathbb{A}_{i,j}) = \frac{2}{qn(n-1)} \leq \frac{1}{n^2}$ for any $q, n > 2$, and for any j coded blocks with $j > 1$, we can construct several disjoint connected graphs to represent the relationship of the underlying plain blocks in the following way. Consider each plain block covered in the j coded blocks as a node, and use an edge to connect two nodes if there is a coded block consisting of the two corresponding plain blocks. We then obtain some disjoint connected graphs with at most one cycle in each graph. The graph with a cycle implies that the plain blocks corresponding with the nodes in this graph can be recovered, since the number of edges is equal to that of nodes, and the set of the coding coefficient

vectors of j blocks is linearly independent.

Next we divide the plain blocks into several disjoint sets according to the graphs, i.e., the plain blocks with the corresponding nodes occurring in the same graph will be put into the same set, and then merge the set with the recoverable blocks, i.e., those blocks represented by the nodes which are in the graph with a cycle. Denote the obtained collection of sets by $\{S_0, S_1, \ldots, S_k\}$, where S_0 contains all the recoverable blocks, and each other set contains some blocks that cannot be recovered. The number of elements in S_i is l_i, and satisfy

$$\begin{gathered} l_i \geq 2, \quad \forall i : 0 \leq i \leq k \\ \textstyle\sum_{i=0}^{k} l_i - k = j \end{gathered}$$

The second equation holds because the number of edges in the graphs corresponding with S_0 is equal to that of nodes, and the number of edges in the graphs corresponding with S_i, $i > 0$, is one less than that of the nodes. At this moment, we can express the probability of $\mathbb{A}_{1,j}$ as follows:

$$\Pr(\mathbb{A}_{1,j}) = \frac{1}{\binom{n}{2}} \left(\binom{l_0}{2} + \frac{1}{q} \sum_{i=1}^{k} \binom{l_i}{2} \right)$$

Since $\binom{a}{2} + \binom{b}{2} < \binom{a+b-1}{2}$, for all $a, b \geq 2$, we further have

$$\begin{aligned} \Pr(\mathbb{A}_{1,j}) &= \frac{1}{\binom{n}{2}} \left(\binom{l_0}{2} + \frac{1}{q} \sum_{i=1}^{k} \binom{l_i}{2} \right) \leq \frac{2}{n(n-1)} \sum_{i=0}^{k} \binom{l_i}{2} \\ &\leq \frac{2}{n(n-1)} \binom{\sum_{i=0}^{k} l_i - k}{2} = \frac{2}{n(n-1)} \binom{j}{2} = \frac{j(j-1)}{n(n-1)} \leq \frac{j^2}{n^2} \end{aligned} \tag{B.3}$$

Given the above equation, in order to show $\Pr(\mathbb{A}_{i,j}) \leq \left(\frac{j}{n}\right)^{2i}$, it suffices to prove that $\Pr(\mathbb{A}_{k,j} | \mathbb{A}_{k-1,j}) \leq \Pr(\mathbb{A}_{1,j})$, which is intuitive to see since given the occurrence of event $\mathbb{A}_{k-1,j}$, the occurrence of event $\mathbb{A}_{k,j}$ implies the k-th block of the first peer is a

linear combination of the j blocks of the second peer, which is identical to the event $\mathbb{A}_{1,j}$. ∎

B.4 Proof of Theorem 4.4

Proof.

$$
\begin{aligned}
&\Pr(\textit{a peer is usable to another peer}) \\
&= \sum_{i=0}^{n} \Pr(p=i) \sum_{j=0}^{n-1} \Pr(p=j)(1-\Pr(\mathbb{A}_{i,j})) \\
&= \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{n} (1-\Pr(\mathbb{A}_{i,j})) \\
&= \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{j} \Pr(\mathbb{A}_{i,j}) \\
&\geq \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \sum_{i=0}^{j} \left(\frac{j}{n}\right)^{2i} \\
&> \frac{n(n+1)}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \left(1 + \frac{j^2}{n^2-j^2}\right) \\
&> \frac{n^2}{(n+1)^2} - \frac{1}{(n+1)^2} \sum_{j=0}^{n-1} \frac{2j^2}{n^2} \\
&= \frac{n^2}{(n+1)^2} - \frac{1}{(n+1)^2} \frac{2n(n-1)(2n-1)}{6n^2} \\
&> \frac{n^2-n}{(n+1)^2} > \frac{(n+1)(n-2)}{(n+1)^2} = \frac{n-2}{n+1}
\end{aligned}
\tag{B.4}
$$

∎

B.5 Proof of Lemma 4.5

Proof. Since there is only one block at each peer, this block can thus be considered to be randomly selected from the whole block collection. We let F_i represent the event that the first i blocks, i.e., the collection of the blocks owned by one of the first i peers, render the content available. Therefore, the probability of content being available given

the block distribution vector p, can be expressed as:

$$\bar{P}_p = \Pr(F_1) + \sum_{i=2}^{l} \Pr(F_i|\bar{F}_{i-1}) \tag{B.5}$$

where $\bar{F}_i$ is the complement of the event F_i. Since $\Pr(F_i) = 0, 1 \leq i < n$, equation (B.5) can be rewritten as:

$$\bar{P}_p = \sum_{i=n}^{l} \Pr(F_i|\bar{F}_{i-1}) \tag{B.6}$$

Since each block can be expressed as $\sum_{i=1}^{n} c_i b_i$, where b_i is the i-th plain block, and c_i is the coding coefficient of b_i, F_i means that the dimension of the space spanned by the set of the coding coefficient vectors of the first i blocks equals n. Denote the coding coefficient vector of the i-th block by C_i, and denote d_i as the dimension of the space spanned by $\{C_k|1 \leq k \leq i\}$. We further let $P_i = \Pr(d_{j+1} = i + 1|d_j = i)$, and let $P'_i = \Pr(d_{j+1} = i|d_j = i)$. Then we have

$$\Pr(F_i|\bar{F}_{i-1}) = \left(\sum_{k=1}^{n-1} P'_k\right)^{i-n} \prod_{j=0}^{n-1} P_j \tag{B.7}$$

Since

$$P_i = \Pr(d_{j+1} = i + 1|d_j = i) = \frac{q^n - q^i}{q^n - 1} \tag{B.8}$$

$$P'_i = \Pr(d_{j+1} = i|d_j = i) = \frac{q^i - 1}{q^n - 1} \tag{B.9}$$

$\Pr(F_i|\bar{F}_{i-1})$ can then be computed by:

$$\begin{aligned}\Pr(F_i|\bar{F}_{i-1}) &= \left(\sum_{k=1}^{n-1} P'_k\right)^{i-n} \prod_{j=0}^{n-1} P_j \\ &= \left(\sum_{k=1}^{n-1} \frac{q^k-1}{q^n-1}\right)^{i-n} \prod_{j=0}^{n-1} \left(1-\frac{q^j-1}{q^n-1}\right) \\ &\geq \left(\frac{\frac{q^n-q}{q-1}-(n-1)}{q^n-1}\right)^{i-n} \left(1-\sum_{j=0}^{n-1}\frac{q^j-1}{q^n-1}\right) \\ &= \left(\frac{1}{q-1}-\frac{n}{q^n-1}\right)^{i-n} \left(1-\left(\frac{1}{q-1}-\frac{n}{q^n-1}\right)\right)\end{aligned} \tag{B.10}$$

Therefore, the probability of content being available is given by

$$\begin{aligned}\bar{P}_p &= \sum_{i=n}^{l} \Pr(F_i|\bar{F}_{i-1}) \\ &\geq \sum_{i=n}^{l} \left(1-\left(\frac{1}{q-1}-\frac{n}{q^n-1}\right)\right)\left(\frac{1}{q-1}-\frac{n}{q^n-1}\right)^{i-n} \\ &= 1-\left(\frac{1}{q-1}-\frac{n}{q^n-1}\right)^{l-n+1}\end{aligned} \tag{B.11}$$

∎

B.6 Proof of Theorem 4.6

Proof. In both cases, there are $\sum_{i=1}^{l} p_i$ blocks in the swarm. We consider the blocks are generated sequentially, which has no impact on the probability of content being available. Denote the dimension spanned by the set of coding coefficient vectors of the first generated i blocks in the case corresponding with p by d_i, and denote the counterpart in the case corresponding with p' by d'_i. We then have

$$\sum_{i=k}^{n} \Pr(d_j = i) \geq \sum_{i=k}^{n} \Pr(d'_j = i), 0 \leq k \leq n, 0 \leq j \leq \sum_{i=0}^{l} p_i \tag{B.12}$$

We prove this equation by induction on the number of blocks generated. Since when there is no block in the swarm, $\Pr(d_0 = 0) = \Pr(d'_0 = 0) = 1$, and equation (B.12) is satisfied trivially. Assume equation (B.12) is true when there are i blocks generated in both cases. When $(i+1)$-th block is generated, we can show that

$$\Pr(d_{i+1} = t | d_i = t) = \frac{q^t - q^{i-\sum_{j=1}^{k} p_j}}{q^n - q^{i-\sum_{j=1}^{k} p_j}}$$

$$\Pr(d'_{i+1} = t | d'_i = t) = \frac{q^t - 1}{q^n - 1}$$

$$\Pr(d_{i+1} = t | d_i = t) + \Pr(d_{i+1} = t+1 | d_i = t) = 1$$

$$\Pr(d'_{i+1} = t | d'_i = t) + \Pr(d'_{i+1} = t+1 | d'_i = t) = 1$$

where k satisfies $\sum_{j=1}^{k} p_j \leq i < \sum_{j=1}^{k+1} p_j$, and it is thus easy to see that $\Pr(d_{i+1} = t+1 | d_i = t) \geq \Pr(d'_{i+1} = t+1 | d'_i = t)$ for all t. Furthermore, for each $k \geq 1$, we have

$$\sum_{j=k}^{n} \Pr(d_{i+1} = j) = \sum_{j=k}^{n} \Pr(d_i = j) + \Pr(d_i = k-1)\Pr(d_{i+1} = k | d_i = k-1)$$

and

$$\sum_{j=k}^{n} \Pr(d'_{i+1} = j) = \sum_{j=k}^{n} \Pr(d'_i = j) + \Pr(d'_i = k-1)\Pr(d'_{i+1} = k | d'_i = k-1)$$

We then discuss the relationship between $\sum_{j=k}^{n} \Pr(d_{i+1} = j)$ and $\sum_{j=k}^{n} \Pr(d'_{i+1} = j)$ in the following two cases: (1) $\Pr(d_i = k-1) \geq \Pr(d'_i = k-1)$, and (2) $\Pr(d_i = k-1) < \Pr(d'_i = k-1)$. In the former case, we have

$$\sum_{j=k}^{n} \Pr(d_{i+1} = j) - \Pr(d'_i = k-1)\Pr(d'_{i+1} = k | d'_i = k-1)$$
$$\geq \sum_{j=k}^{n} \Pr(d_i = j) \geq \sum_{j=k}^{n} \Pr(d'_i = j)$$

In the latter case, we have

$$\begin{aligned}
&\sum_{j=k}^{n} \Pr(d_{i+1}=j) - \Pr(d'_i=k-1)\Pr(d'_{i+1}=k|d'_i=k-1) \\
&\geq \sum_{j=k}^{n} \Pr(d_i=j) + \Pr(d_i=k-1) - \Pr(d'_i=k-1) \\
&\geq \sum_{j=k-1}^{n} \Pr(d'_i=j) - \Pr(d'_i=k-1) \geq \sum_{j=k}^{n} \Pr(d'_i=j)
\end{aligned}$$

Combined with that $\sum_{j=0}^{n} \Pr(d_{i+1}=j) = \sum_{j=0}^{n} \Pr(d'_{i+1}=j) = 1$, we can conclude that

$$\sum_{j=k}^{n} \Pr(d_{i+1}=j) \geq \sum_{j=k}^{n} \Pr(d'_{i+1}=j), 0 \leq k \leq n$$

and thus equation (B.12) has been proven. Based on equation (B.12), we can directly obtain

$$\bar{P}_p = \sum_{k=n}^{n} \Pr(d_{\sum_{i=1}^{l} p_i} = k) \geq \sum_{k=n}^{n} \Pr(d'_{\sum_{i=1}^{l} p_i} = k) = \bar{P}_{p'}$$

∎

B.7 Proof of Theorem 4.7

Proof. We first consider the probability of content being available when network coding is not used, i.e., P_p. To this end, we sort the vector p in non-decreasing order. Assume the resulting vector $p' = [p'_1, p'_2, \ldots, p'_l]$. We then select an element p'_i from p' such that $p'_i < n-1$, and $p'_j = n-1$ for all $j < i$, and construct a new block distribution vector $\hat{p}$, with elements given by

$$\hat{p}_k = \begin{cases} p'_k, & k \neq i, k \neq l \\ p'_i + \min\{p'_l, n-1-p'_i\}, & k = i \\ \max\{0, p'_i + p'_l - n + 1\}, & k = l \end{cases}$$

According to the above equation, we can assume that there are $m - \min\{p'_l, n-1-p'_i\}$

blocks that have been generated, and form the distribution vector $p'' = [p'_1, p'_2, \ldots, p'_{l-1}, \max\{0, p'_l + p'_i - n + 1\}]$. Then we consider the generation of the remaining $r = \min\{p'_l, n - 1 - p'_i\}$ blocks. If the remaining r blocks are generated in peer i, the resulting block distribution vector is $\hat{p}$, and if the last r blocks are generated in peer l, the resulting block distribution vector is p. Let d_i denote the number of different blocks when the i-th block of the remaining r ones is generated. In the former case, the generation of the j-th block of the remaining ones will increment the number of different blocks with the probability

$$\Pr(d_j = k | d_{j-1} = k - 1) = \frac{n - k + 1}{n - (p'_i + j - 1)}$$

and in the latter case, this probability is

$$\Pr(d_j = k | d_{j-1} = k - 1) = \frac{n - k + 1}{n - (\max\{0, p'_l + p'_i - n + 1\} + j - 1)}$$

Therefore, the probability in the first case is larger than that in the second case, and following the proof of equation (B.12) we can derive $P_p = P_{p'} \leq P_{\hat{p}}$. By repeating the construction of new block distribution vector in this way, we finally will get a block distribution vector, $\bar{p}$, with the following elements:

$$\bar{p}_i = \begin{cases} n - 1, & 1 \leq i < \lceil \frac{m}{n-1} \rceil \\ m - (n-1)\left(\lceil \frac{m}{n-1} \rceil - 1 \right), & i = \lceil \frac{m}{n-1} \rceil \\ 0, & i > \lceil \frac{m}{n-1} \rceil \end{cases}$$

It is simple to see that for any other block distribution vector $\breve{p} = [\breve{p}_1, \breve{p}_2, \ldots, \breve{p}_l]$, which satisfies $\sum_{i=1}^{l} \breve{p}_i = m$, we have

$$P_{\bar{p}} \geq P_{\breve{p}}$$

Moreover, it is easy to see that

$$P_{\bar{p}} = 1 - \left(\frac{1}{n}\right)^{c-1} \frac{n - m + c(n-1)}{n} = 1 - (n - m + c(n-1))n^{-c}$$

where $c = \lceil \frac{m}{n-1} \rceil - 1$, and according to equation (4.4) and (4.5), we have $\bar{P}_p \geq 1 - \left(\frac{1}{q-1}\right)^{m-n+1}$. Therefore, we can directly obtain that

$$\frac{1 - P_p}{1 - \bar{P}_p} \geq (n - m + c(n-1))n^{-c}(q-1)^{m-n+1}$$

∎

B.8 Proof of Theorem 4.8

Proof. Assume that the generation of the blocks is in sequential. Let S_i denote the set of coding coefficient vectors of the first i blocks generated, and denote the dimension of the space spanned by S_i by d_i. According to equation (B.3), it is easy to show that if the second coding scheme is used,

$$\Pr(d_{i+1} = t + 1 | d_i = t) \geq 1 - \frac{t^2}{n^2}$$
$$\Pr(d_{i+1} = t + 1 | d_i = t) + \Pr(d_{i+1} = t | d_i = t) = 1$$

and in the original BitTorrent protocol, we have

$$\Pr(d_{i+1} = t + 1 | d_i = t) = 1 - \frac{t}{n}$$
$$\Pr(d_{i+1} = t | d_i = t) = \frac{t}{n}$$

Using the same method in the proof of equation (B.12), it is not difficult to derive equation (4.7). ∎

Bibliography

[1] Ares p2p. http://www.aresp2p.net.

[2] Bittorrent. http://www.bittorrent.com/.

[3] The freenet project. http://freenetproject.org/.

[4] Gnutella. http://www.gnutella.com/.

[5] Kazaa. http://www.kazaa.com/.

[6] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[7] The pirate bay. http://thepiratebay.se.

[8] An estimation of infriging use of the internet. Technical report, Envisional Ltd, January 2011. http://documents.envisional.com/docs/Envisional-Internet_Usage-Jan2011.pdf.

[9] Chdbits. http://chdbits.org, Retrieved September 7, 2011.

[10] R. Ahlswede, Ning Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[11] Nazareno Andrade, Elizeu Santos-Neto, Francisco Brasileiro, and Matei Ripeanu. Resource demand and supply in bittorrent content-sharing communities. *Computer Networks*, 53(4):515–527, March 2009.

[12] David Arthur and Rina Panigrahy. Analyzing bittorrent and related peer-to-peer networks. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 961–969, New York, NY, USA, 2006. ACM.

[13] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving a bittorrent networks performance mechanisms. In *Proceedings of the IEEE INFOCOM 2006 Conference*, Barcelona, Spain, April 2006.

[14] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Some observations on bittorrent performance. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 398–399, New York, NY, USA, 2005. ACM.

[15] Ruchir Bindal, Pei Cao, William Chan, Jan Medved, George Suwala, Tony Bates, and Amy Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *ICDCS'06*, pages 66–75, Lisboa, Portugal, 2006.

[16] Stevens Le Blond, Arnaud Legout, and Walid Dabbous. Pushing bittorrent locality to the limit. *Computer Networks*, 55(3):541 – 557, 2011.

[17] Paul J. Burke. The output of a queueing system. *Operations Research*, 4:699–704, 1956.

[18] Qing-Chao Cai and Kwok-Tung Lo. A detailed survey on a large private bittorrent community. Technical report.

[19] Qingchao Cai, Xiaolu Zhang, and Xuejie Zhang. Streaming live media over bittorrent. In *WRI International Conference on Communications and Mobile Computing*, volume 3, pages 44–49, Kunming, China, January 2009.

[20] D. Carra, G. Neglia, P. Michiardi, and F. Albanese. On the robustness of bittorrent swarms to greedy peers. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2071 –2078, 2011.

[21] J.S.K. Chan, V.O.K. Li, and King-Shan Lui. Performance comparison of schedul-

ing algorithms for peer-to-peer collaborative file distribution. *IEEE Journal on Selected Areas in Communications*, 25(1):146–154, January 2007.

[22] Xiaowei Chen, Yixin Jiang, and Xiaowen Chu. Measurements, analysis and modeling of private trackers. In *The 10th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'10)*, Delft, Netherlands, 2010.

[23] Yuh-Ming Chiu and Do Young Eun. Minimizing file download time in stochastic peer-to-peer networks. *IEEE/ACM Transactions on Networking*, 16:253–266, April 2008.

[24] David R. Choffnes and Fabián E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *Proceedings of the ACM SIGCOMM 2008 conference*, pages 363–374, Seattle, WA, USA, August 2008. ACM.

[25] P. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, 2003.

[26] Alix L. H. Chow, Leana Golubchik, and Vishal Misra. Improving bittorrent: a simple approach. In *Proceedings of the 7th international conference on Peer-to-peer systems*, 2008.

[27] Bram Cohen. Incentives build robustness in bittorrent. In *P2PECON'03*, Berkeley, CA, USA, 2003.

[28] Ruben Cuevas, Nikolaos Laoutaris, Xiaoyuan Yang, Georgos Siganos, and Pablo Rodriguez. Deep diving into bittorrent locality. In *Proceedings of the IEEE INFOCOM 2011 Conference*, Shanghai, China, 2011. IEEE.

[29] C. Dale and Jiangchuan Liu. A measurement study of piece population in bittorrent. In *IEEE GLOBECOM*, pages 405 –410, November 2007.

[30] C. Dana, D. Li, D. Harrison, and C.-N. Chuah. Bass: Bittorrent assisted streaming system for video-on-demand. In *Proceedings of the 7th IEEE Workshop on-Multimedia Signal Processing*, 2005.

[31] Prithula Dhungel, KeithW. Ross, Moritz Steiner, Ye Tian, and Xiaojun Hei. Xunlei: Peer-assisted download acceleration on a massive scale. In *Passive and Active Measurement*, volume 7192 of *Lecture Notes in Computer Science*, pages 231–241. Springer Berlin Heidelberg, 2012.

[32] Kolja Eger, Tobias Hoßfeld, Andreas Binzenhöfer, and Gerald Kunzmann. Efficient simulation of large-scale p2p networks: packet-level vs. flow-level simulations. In *Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*, pages 9–16, Monterey, California, USA, 2007.

[33] Kolja Eger and Ulrich Killat. Bandwidth trading in bittorrent-like p2p networks for content distribution. *Computer Communications*, 31(2):201–211, February 2008.

[34] Bennett Eisenberg. On the expectation of the maximum of iid geometric random variables. *Statistics & Probability Letters*, 78:135–143, February 2008.

[35] Bin Fan, Dah-ming Chiu, and John Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 239–248, Washington, DC, USA, 2006. IEEE Computer Society.

[36] Bin Fan, Dah-Ming Chiu, and John C. S. Lui. Stochastic analysis and file avail-

ability enhancement for bt-like file sharing systems. In *14th IEEE International Workshop on Quality of Service*, pages 30–39, Yale University, USA, 2006.

[37] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, 2004.

[38] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *Proceedings of the IEEE INFOCOM 2005 Conference*, pages 2235–2245, Miami, FL, USA, 2005.

[39] Lei Guo, Songqing Chen, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proceedings of the 2005 Internet Measurement Conference*, pages 35–48, Berkeley, CA, USA, October 2005.

[40] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. A performance study of bittorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169, January 2007.

[41] Jinyoung Han, Taejoong Chung, Hyunchul Kim, T. Kwon, and Yanghee Choi. Systematic support for content bundling in bittorrent swarming. In *IEEE INFOCOM Student Workshop*, San Diego, CA, USA, March 2010.

[42] Jinyoung Han, Taejoong Chung, Seungbae Kim, Hyunchul Kim, Ted Taekyoung Kwon, and Yanghee Choi. An empirical study on content bundling in bittorrent swarming system. *arXiv:1008.2574v1*, 2010.

[43] Jinyoung Han, Taejoong Chung, Seungbae Kim, Ted Taekyoung Kwon, Hyun-Chul Kim, and Yanghee Choi. How prevalent is content bundling in bittorrent. In *Proceedings of the ACM SIGMETRICS joint international conference on Mea-*

surement and modeling of computer systems, pages 127–128, New York, NY, USA, 2011.

[44] Kun Huang, Li'e Wang, Dafang Zhang, and Yongwei Liu. Optimizing the bittorrent performance using an adaptive peer selection strategy. *Future Generation Computer Systems*, 24(7):621–630, 2008.

[45] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Passive and Active Measurements*, Antibes Juan-les-Pins, France, April 2004.

[46] A.L. Jia, X. Chen, X. Chu, and J.A. Pouwelse. From user experience to strategies: how to survive in a private BitTorrent community. Technical Report PDS-2011-004, September 2011. `http://www.pds.ewi.tudelft.nl/fileadmin/pds/reports/2011/PDS-2011-004.pdf`.

[47] A.L. Jia, R. Rahman, T. Vinko, J.A. Pouwelse, and D.H.J. Epema. Fast download but eternal seeding: The reward and punishment of sharing ratio enforcement. In *Proc. IEEE P2P*, pages 280–289, Kyoto, Japan, August 2011.

[48] Seung Jun and Mustaque Ahamad. Incentives in bittorrent induce free riding. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 116–121, Philadephia, PA, USA, August 2005.

[49] Jain Kamal, Lovasz Laszlo, and Chou Philip. Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. *Distributed Computing*, 19(4):301–311, March 2007.

[50] Ian A. Kash, John K. Lai, Haoqi Zhang, and Aviv Zohar. Economics of bittorrent communities. In *Proceedings of WWW*, pages 221–230, New York, NY, USA, 2012.

[51] Sebastian Kaune, Ruben Cuevas Rumin, Gareth Tyson, Andreas Mauthe, Carmen Guerrero, and Ralf Steinmetz. Unraveling bittorrent's file unavailability: Measurements and analysis. In *the 10th IEEE International Conference on Peer-to-Peer Computing*, Delft, Netherlands, 2010. IEEE.

[52] R. Kumar and K.W. Ross. Peer-assisted file distribution: The minimum distribution time. In *Proceedings of the 1st IEEE Workshop on Hot Topics in Web Systems and Technologies*, pages 1 –11, November 2006.

[53] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in bittorrent systems. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 301–312, New York, NY, USA, 2007. ACM.

[54] Arnaud Legout, Guillaume Urvoy-Keller, and Pietro Michiardi. Rarest first and choke algorithms are enough. In *Proceedings of the 2006 Internet Measurement conference*, pages 203–216, Rio de Janeiro, Brazil, 2006.

[55] N. Lev-tov, N. Carlsson, Zongpeng Li, C. Williamson, and Song Zhang. Dynamic file-selection policies for bundling in bittorrent-like systems. In *Proceedings of 18th International Workshop on Quality of Service (IWQoS)*, Beijing, China, June 2010.

[56] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *Proceedings of the ACM SIGCOMM 2008 Conference*, pages 243–254, Seattle, WA, USA, August 2008.

[57] Minglu Li, Jiadi Yu, and Jie Wu. Free-riding on bittorrent-like peer-to-peer file sharing systems: Modeling analysis and improvement. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):954 –966, July 2008.

[58] Q.H. Li and John C. S. Lui. On modeling clustering indexes of bt-like systems. In *Proceedings of the 2009 IEEE international conference on Communications*, pages 1292–1297, Piscataway, NJ, USA, 2009. IEEE Press.

[59] Wei-Cherng Liao, Fragkiskos Papadopoulos, and Konstantinos Psounis. Performance analysis of bittorrent-like systems with heterogeneous users. *Performance Evaluation*, 64:876–891, October 2007.

[60] Ma Lingjun, Pui-Sze Tsang, and King-Shan Lui. Improving file distribution performance by grouping in peer-to-peer networks. *IEEE Transactions on Network and Service Management*, 6(3):149 –162, September 2009.

[61] Nikitas Liogkas, Robert Nelson, Eddie Kohler, and Lixia Zhang. Exploiting bittorrent for fun (but not profit). In *Proceedings of IPTPS*, Santa Barbara, CA, USA, February 2006.

[62] Bo Liu, Yi Cui, Yansheng Lu, and Yuan Xue. Locality-awareness in bittorrent-like p2p applications. *IEEE Transactions on Multimedia*, 11:361–371, April 2009.

[63] Fangming Liu, Ye Sun, Bo Li, and Baochun Li. Quota: Rationing server resources in peer-assisted online hosting systems. In *IEEE International Conference on Network Protocols*, pages 103–112, Princeton, New Jersey, USA, 2009.

[64] Zhengye Liu, Prithula Dhungel, Di Wu, Chao Zhang, and Keith W. Ross. Understanding and improving ratio incentives in private communities. In *the 30th IEEE International Conference on Distributed Computing Systems*, pages 610–621, Washington, DC, USA, 2010. IEEE Computer Society.

[65] Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. Uusee: Large-scale operational on-demand streaming with random network coding. In *Proceedings of the IEEE INFOCOM 2010 Conference*, pages 1–9, San Diego, CA, USA, 2010.

[66] Ziqian Liu and Changjia Chen. Modeling bittorrent-like peer-to-peer systems. *IEEE Communications Letters*, 10(7):513 –515, July 2006.

[67] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in bittorrent is cheap. In *Proceedings of HotNets-V*, Irvine, California, USA, November 2006.

[68] Jianming Lv, Xueqi Cheng, Qing Jiang, Jing Ye, Tieying Zhang, Iming Lin, and Lei Wang. Livebt: Providing video-on-demand streaming service over bittorrent systems. In *PDCAT*, pages 501 –508, 2007.

[69] Laurent Massoulie and Milan Vojnovic. Coupon replication systems. *IEEE/ACM Transactions on Networking*, 16(3):603–616, 2008.

[70] Daniel S. Menasche, Antonio A. A. Rocha, Bin Li, Don Towsley, and Arun Venkataramani. Content availability and bundling in swarming systems. In *the 5th International Conference on Emerging Networking Experiments and Technologies*, pages 121–132, New York, NY, USA, 2009.

[71] Daniel Sadoc Menasche, Rosa M. Meri Leao, Antonio A. A. Rocha, Don Towsley, Edmundo De Souza E, and Arun Venkataramani. Modeling chunk availability in p2p swarming systems. *Performance Evaluation Review*, 37(2):30–32, 2009.

[72] D.S. Menasche, G. Neglia, D. Towsley, and S. Zilberstein. Strategic reasoning about bundling in swarming systems. In *Proceedings of International Conference on Game Theory for Networks*, pages 611 –620, Istanbul, Turkey, May 2009.

[73] M. Meulpolder, L. D'Acunto, M. Capotă, M. Wojciechowski, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips. Public and private bittorrent communities: a measurement study. In *the 9th International Workshop on Peer-to-peer Systems*, San Jose, CA, USA, 2010.

[74] Pietro Michiardi, Krishna Ramachandran, and Biplab Sikdar. Modeling seed scheduling strategies in bittorrent. In *Proceedings of the 6th international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet*, pages 606–616, Berlin, Heidelberg, 2007. Springer-Verlag.

[75] Giovanni Neglia, Honggang Zhang, Don Towsley, Arun Venkataramani, and John Danaher. Availability in bittorrent systems. In *Proceedings of the IEEE INFOCOM 2007 Conference*, pages 2216–2224, Anchorage, Alaska, USA, 2007.

[76] Christian Ortolf, Christian Schindelhauer, and Arne Vater. Paircoding: Improving file sharing using sparse network codes. In *the 2009 Fourth International Conference on Internet and Web Applications and Services*, pages 49–57, Washington, DC, USA, 2009.

[77] John S. Otto, Mario A. Sánchez, David R. Choffnes, Fabián E. Bustamante, and Georgos Siganos. On blind mice and the elephant: understanding the network impact of a large distributed system. In *Proceedings of the ACM SIGCOMM 2011 conference*, pages 110–121, Toronto, Ontario, Canada, August 2011. ACM.

[78] K.N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson. Insights on media streaming progress using bittorrent-like protocols for on-demand streaming. *IEEE/ACM Transactions on Networking*, 20(3):637 –650, june 2012.

[79] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent? In *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, pages 1–14, Cambridge, MA, USA, 2007.

[80] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of*

the 5th USENIX Symposium on Networked Systems Design and Implementation, pages 1–14, Berkeley, CA, USA, 2008. USENIX Association.

[81] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The Bittorrent P2p File-Sharing System: Measurements And Analysis. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, Ithaca, New York, USA, 2005.

[82] Dongyu Qiu and Weiqian Sang. Global stability of peer-to-peer file sharing systems. *Comput. Commun.*, 31:212–219, February 2008.

[83] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of the ACM SIGCOMM 2004 conference*, pages 367–378, Portland, Oregon, USA, August 2004.

[84] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM 2001 conference*, pages 161–172, San Diego, CA, USA, August 2001. ACM.

[85] Shansi Ren, Enhua Tan, Tian Luo, Songqing Chen, Lei Guo, and Xiaodong Zhang. Topbt: a topology-aware and infrastructure-independent bittorrent client. In *Proceedings of the IEEE INFOCOM 2010 Conference*, pages 1523–1531, San Diego, CA, USA, 2010.

[86] Frank Roijers, Michel Mandjes, and Hans van den Berg. Analysis of congestion periods of an m/m/∞-queue. *Performance Evaluation*, 64(7-8):737–754, 2007.

[87] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, Heidelberg, Germany, November 2001. Springer-Verlag.

[88] Hendrik Schulze and Klaus Mochalski. Internet study 2008/2009. Technical report, ipoque. `http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf`.

[89] Improving VoD server efficiency with bittorrent. Choe, yung ryn and schuff, derek l. and dyaberi, jagadeesh m. and pai, vijay s. In *Proceedings of the 15th international conference on Multimedia*, pages 117–126, Augsburg, Germany, 2007.

[90] P. Shah and J.-F. Paris. Peer-to-peer multimedia streaming using bittorrent. In *IPCCC*, pages 340–347, 2007.

[91] Florian Simatos, Philippe Robert, and Fabrice Guillemin. A queueing system for modeling a file sharing principle. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 181–192, New York, NY, USA, 2008. ACM.

[92] Michael Sirivianos, Jong Han, Park Rex, and Chen Xiaowei Yang. Free-riding in bittorrent networks with the large view exploit. In *Proceedings of IPTPS*, Bellevue, WA, USA, February 2007.

[93] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM 2001 conference*, pages 149–160, San Diego, CA, USA, August 2001. ACM.

[94] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 2006 Internet measurement conference*, pages 189–202, New York, NY, USA, 2006. ACM.

[95] Ye Sun, Fangming Liu, Bo Li, Baochun Li, and Xinyan Zhang. Fs2you: Peer-assisted semi-persistent online storage at a large scale. In *Proceedings of the IEEE INFOCOM 2009 Conference*, pages 873 –881, Rio de Janeiro, Brazil, April 2009.

[96] Riikka Susitaival, Samuli Aalto, and Jorma T. Virtamo. Analyzing the dynamics and resource usage of p2p file sharing by a spatio-temporal model. In *Proceedings of the International Conference on Computational Science*, pages 420–427, University of Reading, UK, 2006.

[97] Ye Tian, Di Wu, and Kam Wing Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. In *Proceedings of the IEEE INFOCOM 2006 Conference*, Barcelona, SPAIN, 2006. IEEE.

[98] Gustavo De Veciana and Xiangying Yang. Fairness, incentives and performance in peer-to-peer networks. In *Proceedings of the Forty-first Annual Allerton Conference on Communication, Control and Computing*, pages 150–164, Monticello, IL, 2003.

[99] A. Vlavianos, M. Iliofotou, and M. Faloutsos. Bitos: Enhancing bittorrent for supporting streaming applications. In *Proceedings of the IEEE INFOCOM 2006 Conference*, 2006.

[100] Mea Wang and Baochun Li. R^2: Random push with random network coding in live peer-to-peer streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1655–1666, 2007.

[101] Chi-Jen Wu, Cheng-Ying Li, and Jan-Ming Ho. Improving the download time of bittorrent-like systems. In *ICC'07*, pages 1125 –1129, Glasgow, Scotland, 2007.

[102] Chao Zhang, P. Dhungel, Di Wu, Zhengye Liu, and K.W. Ross. Bittorrent darknets. In *Proceedings of the IEEE INFOCOM 2010 Conference*, San Diego, CA, USA, 2010.

[103] Song Zhang, N. Carlsson, D. Eager, Zongpeng Li, and A. Mahanti. Towards a dynamic file bundling system for large-scale content distribution. In *Proceedings*

of the 19th IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, pages 472 –474, Raffles Hotel, Singapore, July 2011.

[104] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.