

ABSTRACT

CHIU, YUH-MING. On the Performance of Peer Selection Strategies in Stochastic Peer-to-Peer Networks. (Under the direction of Professor Do Young Eun).

Peer-to-peer (P2P) file-sharing applications are becoming increasingly popular and account for a large portion of the Internet's bandwidth usage. Measurement studies show that a typical download session lasts from hours up to several days depending on the level of network congestion or the service capacity fluctuation. In this thesis, we first consider two major factors that have significant impact on the average download time, namely, the spatial heterogeneity of service capacities in different source peers and the temporal fluctuation in service capacity of a given single source peer. We point out that the common approach of analyzing the average download time, or more generally the performance of peer-to-peer networks based on average service capacity is fundamentally flawed. We rigorously prove that both spatial heterogeneity and temporal correlations in service capacity increase the average download time in P2P networks.

We then analyze the impact of the interaction and resource competition between peers on the file download performance under stochastic, heterogeneous, unstructured P2P settings. We introduce the notion of system utilization tailored to a P2P network so as to capture the characteristics of the average download time in a P2P network with multiple competing downloading peers. We then derive an achievable lower bound on the average download time and propose a distributed algorithm with which peers can achieve this minimum average download time, thereby bypassing the curse of spatial heterogeneity and temporal stochastic fluctuation. Our algorithm relies on constantly changing connected source peers and selecting source peers probabilistically. The performance of different peer selection algorithms is compared under NS-2 simulations. Our results also provide theoretical explanation to the inconsistency of performance improvement by using parallel connections (parallel connection sometimes does not outperform single connection) observed in some measurement studies.

On the Performance of Peer Selection Strategies in Stochastic Peer-to-Peer Networks

by
Yuh-Ming Chiu

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2009

APPROVED BY:

Dr. Mihail Sichitiu

Dr. Khaled Harfoush

Dr. Do Young Eun
Chair of Advisory Committee

Dr. Michael Devetsikiotis

DEDICATION

To my parents, Sheng-Sheng Chiu and Chiu-Hua Wang,
and my wife, Ming-Ying Chou.

謹獻給我的父母：邱盛生與王九華
以及我的妻子：周明瑩

BIOGRAPHY

Yuh-Ming Chiu received his B.S. degree from the Department of Communication Engineering, National Chiao Tung University, Taiwan, in 1997, and his M.S. degree from the Department of Electrical Engineering, National Tsing Hua University, Taiwan, in 2000. He has been a graduate student in the Department of Electrical and Computer Engineering, North Carolina State University since 2004. Before joining the doctoral program, he had worked with Siscomm (Taiwan) and Accton (Taiwan) each for 2 years. His research interests include queueing analysis and performance analysis of peer-to-Peer networks. He has been a member of IEEE since 2006.

ACKNOWLEDGMENTS

First, my deepest gratitude goes to my advisor, Professor Do Young Eun, for his training and guidance. Professor Eun always gives me constructive comments and sharp insights to my research problems no matter how trivial my ideas seem to be. He always encourages his students to “think outside of the box” and look at problems from as many different perspectives as possible. He demonstrated and taught me how to be a genuine researcher during the course of my study. I am grateful and privileged to have such a patient, inspiring, and encouraging advisor. I would also like to thank Professor Michael Devetsikiotis, Professor Mihail Sichitiu, and Professor Khaled Harfoush for serving on my committee and providing their invaluable advise and comments on my research topic.

Second, I was very fortunate to have Dr. XinBing Wang, Han Cai, SungWon Kim, ChulHo Lee, and Daehyun Ban as my colleagues. I enjoyed and benefited very much from having enlightening technical discussions with them. I am grateful and greatly appreciate their friendship.

Last but not the least, I want to thank my parents, Sheng-Sheng Chiu and Chiu-Hua Wang for their unconditional love and support. My parents always stood by me in the ups and downs of my life and they provided words of wisdom when I was facing difficult decisions. Words are not enough to describe my gratitude and appreciation for them. It is a blessing to have Ming-Ying Chou as my wife, who sacrificed her career opportunity to have a family with me. Without her full support, I could not have finished this thesis on time.

TABLE OF CONTENTS

LIST OF TABLES.....	vii
LIST OF FIGURES	viii
1 Introduction.....	1
1.1 Motivation	3
1.1.1 Impact of Stochastic Characteristics of Service Capacity	3
1.1.2 Impact of Competition and Parallel Connections	6
1.2 Contributions	7
1.3 Outline	9
2 Preliminaries on Peer-to-Peer Networks	10
2.1 Definition of P2P	10
2.2 P2P Content Distribution	12
2.2.1 Data Location and Information Routing	14
2.2.2 Data Replication and Peer Selection Strategies	19
2.2.3 Reputation and Incentive Mechanisms	21
3 Uniform Peer Selection Strategies in a Stochastic Environment	23
3.1 Introduction and Background	23
3.1.1 Heterogeneity of Service Capacity	23
3.1.2 Correlations in Service Capacity	24
3.2 Characterizing the Download Time in a P2P Network	25
3.2.1 Impact of Heterogeneity in Service Capacity	26
3.2.2 First Hitting Time of a Correlated Stochastic Process	27
3.2.3 First Hitting Time and Degree of Correlation	30
3.3 Minimizing Average Download Time Over Stochastic Channels	32
3.3.1 Random Chunk-based Switching	34
3.3.2 Random Periodic Switching	35
3.3.3 Discussion	40
3.4 Numerical Results	41
3.4.1 Single Downloading Peer with Heterogeneous Service Capacities . .	41
3.4.2 Multiple Downloading Peers with Competition	45
4 Non-Uniform Selection Under Competition and Parallel Connections...	50
4.1 System Model with Multiple Downloading Peers	50
4.2 Download Time Analysis	52
4.2.1 System Utilization	52
4.2.2 Achievable Minimum Average Download Time	55

4.3	Achieving the Minimum Average Download Time	58
4.3.1	Dynamic Peer Selection	58
4.3.2	Impact of Parallel Connections	63
4.3.3	Maximizing System Utilization	65
4.3.4	Optimal Peer Selection Example	67
4.3.5	Adaptive Peer Selection Strategy	68
4.4	Simulations	71
5	Conclusions and Future Work.....	77
5.1	Conclusion	77
5.2	Future Work	78
	Bibliography	81

LIST OF TABLES

Table 3.1	Average service capacity of each source peer under different configurations. .	42
-----------	--	----

LIST OF FIGURES

Figure 2.1 Data location and retrieval under Gnutella protocol.....	15
Figure 2.2 Normal KaZaa peers connect to Supernodes whereby the search is routed through the Supernodes and downloads are performed directly between two peers.	16
Figure 2.3 Data location service is performed by a centralized server (tracker) in BitTorrent.	17
Figure 2.4 Chord ring with identifier circle consisting of ten peers and five data keys. It shows the path followed by a query originated at peer 8 for the lookup of key 54 without the help of Finger table and the Finger table entries for peer 8.....	18
Figure 3.1 Typical variation in end-to-end available bandwidth based on the results in [89, 90]. Drastic changes usually occur in the scale of minutes.....	24
Figure 3.2 Relationship between the average download time and different degrees of correlation ρ	31
Figure 3.3 The operation of source selection function $U(t)$ for random periodic switching	36
Figure 3.4 Average download time vs. degree of heterogeneity under different download strategies and different degree of correlations.	43
Figure 3.5 Performance comparison of different strategies under different levels of competition.	47
Figure 4.1 The relation between the average download time $\mathbb{E}\{T\}$, system utilization ρ , and the level of competition ν for different average number of parallel connections L . $\mathbb{E}\{T\}$ is normalized over $F/A(\vec{c})$	63
Figure 4.2 The optimal connection probabilities to source peers (from (4.33)) offering different service capacities.....	67
Figure 4.3 A random selected downloading peer i estimates the level of competition in first two time slots.	69
Figure 4.4 The illustration of the network used in our NS-2 simulation. We assume that the Internet backbone does not introduce any congestion. The upstream access link of each peer is bandwidth limited.....	72

- Figure 4.5 The average download time for periodic uniform peer selection and “connect-and-wait” strategy. There are 50 nodes and no free-riders, hence $|\mathcal{S}| = 50$. The number of downloading peers varies from 10 to 50. 74
- Figure 4.6 The average download time for three dynamic peer selection strategies: periodic uniform peer selection (Uniform), the centralized optimal strategy in (4.33)–(4.34) (Optimal), and the distributed strategy in Algorithm 1 (Distributed). 75
- Figure 4.7 The reduction in the average download time for different peer selection strategies over the uniform peer selection. 76

Chapter 1

Introduction

The emergence of “Peer-to-Peer” (P2P) network technologies has attracted much social and technological attention and brought new challenges to research communities over the past decade. Although the use of P2P technologies can be traced back to a much earlier time, it was not until recent years that we see a large variety of P2P network applications. In addition to the well known file-sharing applications such as BitTorrent, Gnutella, and eMule [1, 2, 3], we see P2P streaming applications as well, such as CoolStream, Tribler, SopCast, just to name a few [4, 5, 6]. Further, P2P technologies are applied in the area of telecommunication, such as Skype [7], or in the area of academic and scientific computing, such as SETI@home or sciencenet-YaCy [8, 9]. Recent measurement studies have found that the volume of P2P traffic accounts for approximately 70% of the total broadband traffic, which surpasses the traffic volume generated by many other application protocols such as E-mail, FTP, and HTTP [10, 11, 12, 13].

P2P technology in fact offers some promising advantages over the traditional “client-server” networking model from a technical point of view, despite that some think the popularity of P2P applications only comes from the piracy of copyrighted materials. A peer-to-peer network is inherently distributed and scalable because each node (peer) in the network contributes its resources to other peers while consuming resources from others. In other words, each peer in the network acts both as a server and as a client at the same time. In a P2P network, even nodes at the edge of the network, i.e. home computers that are often considered as pure clients that only consumes resources, contribute their resources to other peers. Hence, the aggregated resource available in the network scales with the size

of the network [14, 15, 16]. On the other hand, under the client-server model, the amount of resources in the network is determined solely by the limitation of hardware and software specifications of the few servers in the network. Ideally, the “equal share” of resources that each peer can receive from a P2P network is much larger than that of a client-server network. Therefore, one can expect that the performance of P2P networks ought to outperform that of client-server networks in various applications.

Unfortunately, the performance of a P2P network often does not match people’s expectations. Taking file-sharing applications for example, real world measurement results show that downloading a 100MB file from a P2P network may take days to weeks to complete [17, 18] whereas it would take approximately 4 hours for a dedicated FTP server to transfer a file of the same size to one of its clients. Why do we have such seemingly counter-intuitive results? There are many reasons for a P2P network not achieving expected performance. Studies have shown that most nodes in real world file-sharing P2P networks are personal computers (PC) [17, 18]. We list below some characteristics that differentiate ordinary PCs from corporate operated server machines .

1. The hardware and software specifications between different computers varies in a wide range.
2. A peer can go online/offline at any time.
3. The owner of the machine may not want to share the resource of the machine.

The second and the third item above are actually results of human behavior. For example, we often turn off our home computers when we are not using them or when we go to work. We simply close or lower the execution priority of unnecessary applications if we want our computer to dedicate to more important jobs. In fact, the quality of resource each peer receives from the network depends on the quality of service that its “neighboring peers” can offer. In other words, the amount or the quality of resources *is not equally distributed among all peers*. Therefore, peer selection strategy plays an important role in determining the performance a peer receives from the network [19, 20, 21, 22, 23, 24, 25, 26, 27].

In addition to the issue of peer selection, there are other research challenges regarding the performance of a peer-to-peer network. For example, some peers are free-riders – peers who only consume system resources but never contribute – and encouraging peers

to contribute to the network is a topic under active research [28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Further, data search mechanisms and the how a P2P network organizes itself are also very important issues [38, 39, 40, 41, 42, 43, 44, 45, 46, 47]. Due to the distributed nature of P2P network, resources are often scattered around the entire network. When a peer joins a P2P network, it has to search for the set of peers that can offer the required service. A well designed peer organization method or search algorithm can help each peer find what it needs efficiently. More detailed description on challenges facing P2P technologies is provided in Chapter 2.

1.1 Motivation

It is impossible to address all research issues related to P2P networks and its applications in one thesis. Instead, we focus on the performance of peer selection strategies in P2P content distribution applications in this work. Note that file-sharing and streaming are the two most popular applications that utilize P2P technology. File download time is an important performance metric in these applications. It is intuitive that a user of the network will be more satisfied if he/she can download a file from the network faster. In streaming applications, file download time implies the filling time of the playback buffer. The time required to fill up the playback buffer determines the playback delay and potentially the continuity of playback after the peer experiences service interruption. Hence, reducing file download time improves the user experience of P2P networks.

1.1.1 Impact of Stochastic Characteristics of Service Capacity

Recall from the beginning of this chapter, most analysis on the file download time is based on the assumption that the aggregated capacity of the entire network being *equally distributed* among all peers that are downloading the same file. At the same time, participating peers contribute the same amount of resource, i.e. capacity, to the network, over time [14, 15, 48, 16, 49, 50, 51]. In reality, peers can have different hardware and software specifications. The access bandwidths of different peers can be different. The wide range of variations in the capacity of different peers suggest that the network is a heterogeneous network. Further, most peers in a P2P network are home computers, which imply that they are less likely to offer capacity or resources with constant quality. The

capacity offered by each peer can fluctuate with time. The common practice used for download time analysis which *based on the averaged quantities* [52, 20, 53, 14, 15, 21, 54], e.g., average capacities of all possible source peers in the network may give misleading results. We illustrate limitations of the approach based on averaged quantities in a P2P network by considering the following examples.

Limitations of Approach via Average Service Capacity

Suppose that a downloading peer wants to download a file of size F from N possible source peers. Let c_i be the average end-to-end available capacity between the downloading peer and the i^{th} source peer ($i = 1, 2, \dots, N$). Notice that the actual value of c_i is *unknown* before the downloading peer actually connects to the source peer i . Hence, the best strategy for a downloading peer is select one of the possible peers uniformly at random. Uniform selection is a quite common strategy in P2P networks [48, 50, 51]. The average service capacity of the network, \bar{C} , is give by $\bar{C} = \sum_{i=1}^N c_i/N$. Intuitively, the average download time, T , for a file of size F would be

$$T = F/\bar{C}. \quad (1.1)$$

In reality, however, (1.1) is far different from the true average download time for each user in the network. The two main reasons to cause the difference are (i) the *spatial heterogeneity* in the available service capacities of different end-to-end paths and (ii) the *temporal correlations* in the service capacity of a given source peer. We first consider the impact of heterogeneity. Suppose that there are two source peers with service capacities of $c_1 = 100\text{kbps}$ and $c_2 = 150\text{kbps}$, respectively, and there is only one downloading peer in the network. Because the downloading peer does not know the service capacity of each source peer¹ prior to its connection, the best choice that the downloading peer can make to minimize the risk is to choose the source peers with equal probability. In such a setting, the average capacity that the downloading peer expects from the network is $(100 + 150)/2 = 125\text{kbps}$. If the file size F is 1MB, we predict that the average download time is 64 seconds from (1.1). However, the actual average download time is

¹Although the fluctuation seen by a downloading peer can be caused by change both in the status of the end-to-end network path and in the status of the source peer itself, we use “service capacity of a source peer” to unify the terminology throughout this thesis.

$1/2(1\text{MB}/100\text{kbps}) + 1/2(1\text{MB}/150\text{Mbps}) = 66.7$ seconds! Hence, we see that the spatial heterogeneity actually makes the average download time longer.

Suppose now that the average service capacity can be known *before* the downloading peer makes the connection. Then, an obvious solution to the problem of minimizing the average download time is to find the peer with the *maximum average capacity*, i.e., to choose peer j with the average capacity c_j ($c_j \geq c_i$ for all i), as the average download time T_i over source peer i would be given by F/c_i . We assume that each peer can find the service capacity of its source peers via packet-level measurements or short-term in-band probing [55].

Consider again the previous two-source peer example with $c_1 = 100\text{kbps}$ and $c_2 = 150\text{kbps}$. As we want to minimize the download time, an obvious choice would be to choose source peer 2 as its average capacity is higher. Now, let us assume that the service capacity of source peer 2 is not a constant, but is given by a stochastic process $C_2(t)$ taking values 50 or 250kbps with equal probability, thus giving $\mathbb{E}\{C_2(t)\} = c_2 = 150\text{kbps}$. If the process $C_2(t)$ is strongly correlated over time such that the service capacity for a file F is likely to be the same throughout the session duration, it takes on average $(1\text{MB}/50\text{kbps} + 1\text{MB}/250\text{kps})/2 = 96$ seconds, while it takes only 80 seconds to download the file from source peer 1. In other words, it may take longer to complete the download when we simply choose the source peer with the maximum average capacity! It is thus evident that the impact of correlations (second-order statistics) or higher-order statistics associated with the capacity fluctuation in time will need to be taken into account, even for finding a source peer with minimum *average* download time.

From the simple example above, we can see that both the heterogeneity and the temporal stochastic fluctuation in service capacity plays an important role in determining the file download time. Although the downloading peer selects its source peer with equal probability, the average download time is much longer than *the file size divided by the average received capacity!* Then, it is natural to ask whether equation (1.1) is achievable in reality and under what condition we can have (1.1). We discuss this question in great detail in Chapter 3.

1.1.2 Impact of Competition and Parallel Connections

The simple example of a single downloading peer in the previous section demonstrates how spatial heterogeneity and temporal correlation in service capacities of the peers affect the file download performance. If a downloading peer is not able to get the information about the service capacity of its source peers, uniform random selection is the best a downloading peer can do. On the other hand, some protocols, such as Gnutella, exchange the maximum upload bandwidth (the physical access bandwidth) information among peers. Further, a downloading peer may probe the available bandwidth of each possible source peer before it make a decision. Therefore, a downloading peer may be able to receive better performance by non-uniform peer selection based on the information about other peers' upload bandwidth. There are several existing results in the literature suggesting how a downloading peer should select its source peers if the information about the upload bandwidth of each peer is known. However, all the results in [19, 20, 21] have been established under the assumption that *there is only one downloading peer in the network*. This is critical, since in a real P2P network there will be multiple peers uploading and downloading at the same time and a peer's service capacity will be shared by its competing peers. In other words, the downloading peers will have to compete for the limited resource each single source peer can offer. In this setting, the download performance is determined not only by the stochastic fluctuation and heterogeneity in service capacity that each peer offers; how each peer makes its peer selection choice under such a competitive environment is also very important.

Further, it is often believed that parallel connection can help a downloading peer to receive more capacity from the network. It is obvious that a downloading peer should utilize as many connections as possible. However, real world measurements show some rather counter-intuitive results [56, 52]. The results in [56] show that if some peers utilizing parallel connections while others don't, then the ones utilizing parallel connections have shorter average download time. However, if every downloading peer utilizes parallel connections, then the number of connections does not impact performance much. In other words, the performance of utilizing utilizing few number (3 or 4) of parallel connections is almost the same as utilizing large number of parallel connections (20 ro 100). It is shown in [52] that even utilizing a single connection performs as good as utilizing parallel connections. Based on the counter-intuitive measurement results, it is interesting to know under what condition

parallel connection is beneficial. In Chapter 4, the performance of utilizing non-uniform peer selection and parallel connections is extensively analyzed.

1.2 Contributions

We summarize the contributions of the work in this thesis as follows.

- First, we start with analyzing a simple case in which there is only one downloading peer in the network. We show that the predicted value given in (1.1) in Section 1.1.1 is actually achievable regardless of the distribution of service capacities and correlations in a P2P network.

We first characterize the relationship between the heterogeneity in service capacity and the average download time for the downloading user, and show that the degree of diversity in service capacities has negative impact on the average download time. After we formally define the download time over a stochastic capacity process, we prove that the correlations in the capacity make the average download time much larger than the commonly accepted value F/c , where c is the average capacity of the source peer. It is thus obvious that the average download time will be reduced if there exists a (possibly distributed) algorithm that can efficiently eliminate the negative impact of both the heterogeneity in service capacities over different source peers and the correlations in time of a given source peer.

In practice, most P2P applications try to reduce the download time by minimizing the risk of getting stuck with a ‘bad’ source peer (the connection with small service capacity) by using smaller file sizes and/or having them downloaded over different source peers (e.g., parallel download).² In other words, they try to reduce the download time by minimizing the *bytes* transferred from the source peer with small capacity. However, we show that this approach cannot effectively remove the negative impact of both the correlations in the available capacity of a source peer and the heterogeneity in different source peers. This approach may help to reduce average download time in some cases but not always. Rather, a simple and distributed algorithm that

²For example, Overnet, BitTorrent, and Slurpie divide files into 9500KB, 256KB, and 256KB file segments (chunks), respectively [57, 58, 1], and a downloading peer can transfer different chunks from different source peer.

limits the amount of *time* each peer spends on a bad source peer, can minimize the average download time for each user almost in all cases as we will show later in Chapter 3. Through extensive simulations, we also verify that the simple download strategy outperforms all other schemes widely used in practice under various network configurations. In particular, both the average download time and the variation in download time of our scheme are smaller than any other scheme when the network is heterogeneous (possibly correlated), as is the case in reality.

- Second, assume that there are multiple downloading peers in the network and each peer can make peer selection decision based on the knowledge about the maximum upload capacity that its neighbors can offer. In this case, we not only have to consider the network heterogeneity and the stochastic characteristics of the service capacity that each source peer can offer but also the impact of interaction/competition among the downloading peers.

In order to capture this impact of interactions among the downloading peers, we introduce the notion of system utilization – defined as the average fraction of the aggregated system capacity that is *actually* consumed by all the downloading peers, which is heavily dependent upon the peer selection decision. With the introduction of system utilization, we show how the average “share” of system capacity a downloading peer receives is not always equal to the aggregated capacity of the source peers divided by the number of downloading peers, *even if each source peer distributes its capacity evenly among its connected downloading peers*. Further, we extend the results in [25] that the average file download time is often greater than the file size divided by the average received capacity of each downloading peer in our much more complex network setting. We derive an *achievable* lower bound on the average download time based on the average network capacity, the system utilization, and the level of competition in the network.

It was shown in [25] that switching a single connection “periodically” among possible source peers “uniformly at random” can reduce the average download time to some extent. In a network with *multiple downloading peers*, we show that the algorithm in [25] is far from being optimal, but can still be used as the first step toward developing an optimal peer selection strategy. We modify the notion of periodic switching

and start by letting each downloading peer generate *a random number of parallel connections periodically* instead of having just one single connection all the time. Each downloading peer chooses its source peers according to some probability. We can then formulate the probability assignment problem as a convex optimization problem and compute the optimal connection probabilities in a centralized fashion. Our solution is applicable to much wider ranges of network scenarios with stochastic fluctuation and heterogeneity under peer competitions, to which none of the approaches in [25, 19, 20, 21] applies.

While the aforementioned centralized algorithm to solve for the optimal peer selection probability is impractical in a fully distributed P2P network, the algorithm will serve its purpose as a benchmark with optimal performance. We then propose a heuristic distributed algorithm that only depends upon the network performance metric *observed by each downloading peer*. Through realistic network simulations using NS-2 [59], we compare the performance of different peer selection strategies. We use the performance of periodic uniform peer selection derived from [25] as the baseline for performance comparison. Our NS-2 simulation results show that both the optimal centralized algorithm and the proposed distributed algorithm outperforms the periodic uniform peer selection in all cases. Our centralized algorithm can even reduce the average download time by 50% compared to periodic uniform peer selection. In addition, our results also provide theoretical explanation for the observations in [56, 52] that using parallel connections does not always give users better performance than using a single connection. We present conditions under which parallel connections is helpful and provide a guideline for determining a suitable number of parallel connections.

1.3 Outline

This thesis is organized as follows. Chapter 2 describes the background and current advancement of P2P network technologies. Chapter 3 analyzes the performance of different uniform peer selection strategies in a heterogeneous and stochastic P2P network. Chapter 4 analyzes the impact of competition among peers and performance of non-uniform peer selections with each peer utilizing parallel connections. The conclusion and future work are presented in Chapter 5.

Chapter 2

Preliminaries on Peer-to-Peer Networks

Some of the technical advantages of P2P networks include its scalability and reliability against single-point failures compared to the traditional client-server network architecture. This chapter provides some background on P2P network technologies as well as the research challenges facing P2P networks. Interested readers are encouraged to refer to the referenced papers for more detailed information.

2.1 Definition of P2P

The commonly known network architecture is the “client-server” model in which each node (machine) has a very specific role. A “server” offers specific service(s), such as E-mail or data storage, while a client solely consumes or utilizes services from the server. The flow of valuable data (or service) follows a single direction, i.e. from servers to clients. Under the client-server model, the service of a server is shared by all the clients that connect to it. As the number of clients grow, the quality or the “share” of the service that a client receives from the server degrades since a single server always has its physical limitations in terms of computing power, bandwidth, or storage space, although those limits can be very large. Hence, the service capacity of a single server does not scale with the demand. Further, when the servers fails, as what had happened with Google, Yahoo, or Amazon, all clients suffers service interruption and the service providers lose their market opportunities [60].

P2P network technology is one of the approaches that aim to solve the short coming of client-server networks, namely, scalability and fault-tolerance. There is no notion of pure server or pure client under P2P model. Each node in the network is both a server and a client. There is no *centralized service provider* in the network. The resources of each peer, such as computing power, bandwidth, or storage space, are shared with each other. Since each peer contributes its own resources while consume resources from others, the aggregated supply of resources increases with the number of peers in the network. In other words, the service capacity, storage space, and computing power of the entire network *scales with the number of peers in the network* as shown in [14, 15, 16, 49]. The ability to scale the amount of resources with the size of the network is the major characteristic that differentiate a P2P network from traditional client-server network. Note that having all nodes being a client and server at the same time suggest that even home computers can be providing services to other nodes. Therefore, peers in a P2P network are often required to have the ability to tolerate both instability in service quality and intermittent connectivity.

In short, a P2P network is often defined by the the following characteristics [61]:

1. P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet.
2. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers.

Note that the above definition is a very “strict” definition for a peer-to-peer network. Even some of the well known P2P application does not hold all the above characteristics. According to the definition of a P2P system in [61], many applications that rely on the help of centralized servers, such as BitTorrent [1], Napster, or instant message systems, can not be considered as a P2P network under this definition. In this work, we take a broader definition of P2P network given in [62]: “Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.” Actually, this weaker definition is more close to the notion

of a P2P network perceived by the community.

2.2 P2P Content Distribution

Although P2P technologies are used in many areas such as telecommunication and distributed computing, most of the applications falls into the category of content distribution, which includes the systems and infrastructures that support sharing of digital media or data among peers. Basically, a peer-to-peer content distribution system is a distributed storage medium that allows for the publishing, searching, and retrieval of files by members of its network. Those content distribution systems can be further categorized based on the design requirements into the following groups:

- P2P file-sharing applications. The most widely known application of P2P networks. This type of application utilizes P2P technology in its basic form. There is almost no system design requirements for this type of application. The performance of the network is often measured by the file download time of the peers. Due to the scalability of P2P networks, it is often believed that peers can receive better performance than that of client-server networks. Some examples of pure P2P file-sharing applications are Gnutella, KaZaa, and eMule, just to name a few[2, 63, 3].
- P2P streaming applications. These systems are targeted towards the delivery of media contents to the peers in realtime or with very little delay. Streaming services often has minimum bandwidth requirements. In addition, the sequence of “Bytes” at which the media arrives at each peer determines the performance as well. The main focus of such application is the timeliness and the order of the arrival of the content. CoolStream, Tribler, and SopCast are just few examples of P2P streaming applications [4, 5, 6].
- P2P storage applications. These systems are targeted towards treating the entire network as a distributed storage space, in which peers can publish, search, and store files in a *secure and persistent manner*. The content of the medium has to be accessed by a controlled manner by peers with privilege. Hence, the security and the persistency of the contents are of importance. MojoNation, OceanStore, and FreeHaven are few examples of P2P storage applications [64, 65, 66]. Unlike the file-sharing and streaming

application, P2P storage systems have not attract as much attention as file-sharing and streaming systems.

Due to various P2P application requirements and the distribute nature of P2P networks, many application independent network infrastructures and basic network services are developed to serve as the building blocks for P2P systems. Since file-sharing and streaming applications are more popular than P2P storage systems, we will focus our study on those building blocks designed for file-sharing and streaming applications. First, we brief introduce some basic infrastructure services required for file-sharing and streaming as follows:

- *Data location and information routing.* Due to the distributed nature of a P2P network, peers along with the their data are scattered all over the Internet. Further, the peers can join and leave the network at any time. Hence, it is important to have algorithms that help peers to locate their data of interest efficiently with fault-tolerance. A more detailed review of the algorithms in this category is presented in the next section.
- *Data replication and peer selection strategies.* All peers are not created equal! As shown in [17, 18], peers have different hardware and software specifications and their ability to service others are different. How each peer choose its remote peers to retrieve the file can determine their received performance. In the next section, we give a overview of various file replication and peer selection methods that are proposed in the literature.
- *Reputation and incentives mechanisms* In a P2P network, there is no centralized authority to control peers' behavior. It is likely for some peers to act selfishly, i.e. consume the network resources without contributing anything back. Those selfish peers are called "free-riders." Free-riders are a major issue in P2P networks because the scalability of a P2P network relies upon each peer's willingness to share their resources. Therefore, it is important to encourage cooperative behavior among peers and punish those who don't cooperate. Various incentive algorithms based on reputation, credit, or resource trading are developed to combat free-riders.

Next, we present a detailed review of infrastructure services.

2.2.1 Data Location and Information Routing

The operation of any P2P content distribution system relies on a network of peer computers and the connections between them. This network is often formed on top of and independent from the underlying physical computer network, e.g. IP network. An P2P network is also referred to as an “overlay” network. Overlay networks can be distinguished in terms of their centralization and structure. In this section, we present an overview of the data location and information routing techniques under different types of overlay networks

Degree of Centralization

A overlay network can be classified into three types, 1. decentralized, 2. partially decentralized, and 3. centralized networks, based on the degree of centralization. The decentralized overlay networks realize the purest form of P2P network. The functionality and the tasks that each peer performs are identical and there is no distinction between one peer from another. Each peer can issue, relay, and reply to query or service messages. In a partially decentralized P2P network architecture, some peers may have more important role than others, such as acting as a message relay hub or providing local index lookups for its neighboring peers. The partially decentralized overlay network is a natural adaptation of the fully decentralized overlay to the heterogeneous capabilities among different peers. Peers with higher storage space or CPU power are often elected as localized information hubs, and hence are given a more important role. A centralized overlay network suggests that there is a centralized server that provides the data lookup service for the entire network even though the end-to-end interactions and file exchanges are performed directly between two peers. Obviously, the centralized architecture violates the design concept of a P2P network, although this type of overlay network is still sometimes considered as P2P, e.g. BitTorrent. The centralized overlay network is vulnerable to single-point of failure and malicious attacks just like client-server model.

Network Structure

The technical meaning of “structured” is that the P2P overlay network topology is *tightly controlled and contents (or pointers to them) are placed not at random peers* but at specified locations that will make subsequent queries more efficient. Each peer and

data object is assigned a unique index (or address) in the overlay network. These systems essentially provide a mapping between content (e.g. file identifier) and location (e.g. node address), in the form of a distributed routing table (DHT). Whether two peers are able to communicate with each other depends on the address assigned to them. On the other hand, in an unstructured P2P network, the location of the content can be anywhere and there is no restrictions on how two peers connect.

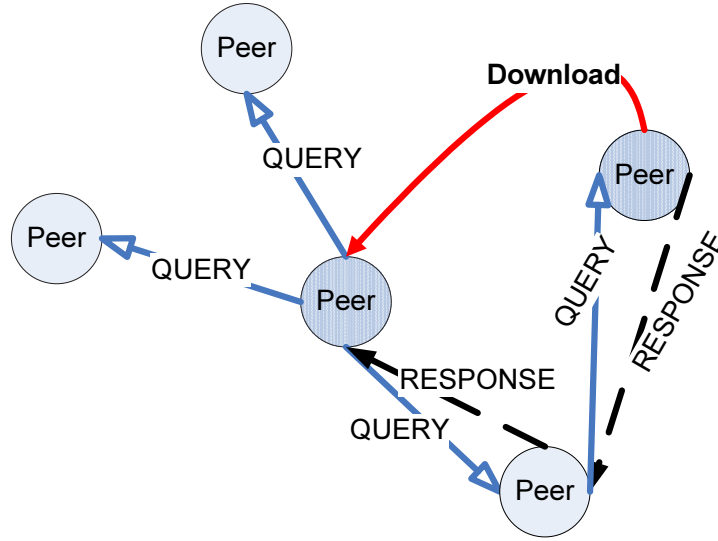


Figure 2.1: Data location and retrieval under Gnutella protocol

Here, we will first discuss the data searching and message routing mechanisms widely used in unstructured networks. In reality, Gnutella, KaZaa, and BitTorrent networks are categorized as unstructured networks. Further, Gnutella, KaZaa, and BitTorrent are representative examples of fully decentralized, partially decentralized and centralized overlay networks, respectively. How data location and information routing are performed in those networks are shown in Figure 2.1, Figure 2.2, and Figure 2.3 (adapted from [67]). From Figure 2.1 and 2.2, we can easily see that the query messages floods the entire network whenever a data location service is performed in unstructured and decentralized systems, namely, Gnutella and KaZaa, thus generating a large amount of overhead traffic.

The authors of [42] propose two algorithms: “expanding ring” and “k-walker random walk” to overcome the disadvantages of pure message flooding. Note that random

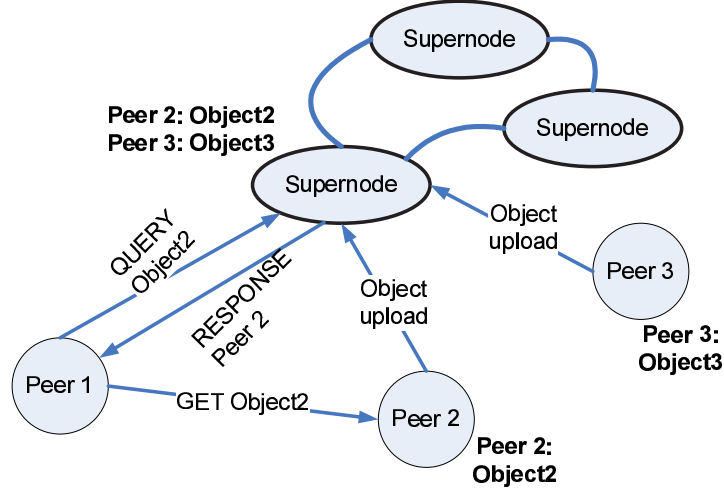


Figure 2.2: Normal KaZaa peers connect to Supernodes whereby the search is routed through the Supernodes and downloads are performed directly between two peers.

walk type of search methods are the most aggressive method in reducing the search traffic. However, studies have shown that random walk type of data location service may have large latency in general network topologies. If the network topology is a power-law graph, i.e. the degree distribution of the nodes follows power-law distribution, then the latency of random walk type of search methods can be much reduced compared to general network topologies [68]. Hence, some research suggest that when a peer joins the network, it should careful select its neighboring peers (ones with direct connections) rather than connecting to the existing peers in ad-hoc fashion so that the resulting unstructured network topology have the desired property [69, 46, 70, 71, 72].

Another approach to solve the problem of the unguaranteed search latency or heavy search traffic in unstructured P2P network is to make it structured. In general, structured P2P networks are often referred to as networks utilizing distributed hash table (DHT), such as CAN, Chord, Pastry, and Tapestry [73, 40, 39, 45]. In a DHT network, nodes and data objects are each assigned a unique identifier (key). Each peer maintains a small routing table consisting of its neighboring peers' node id and IP addresses. Lookup queries or messages are forwarded across overlay paths to peers in a progressive manner in which the node identifier after each hop gets closer to the the identifier of the data object. Different DHT-based systems have different organization schemes for the data objects and

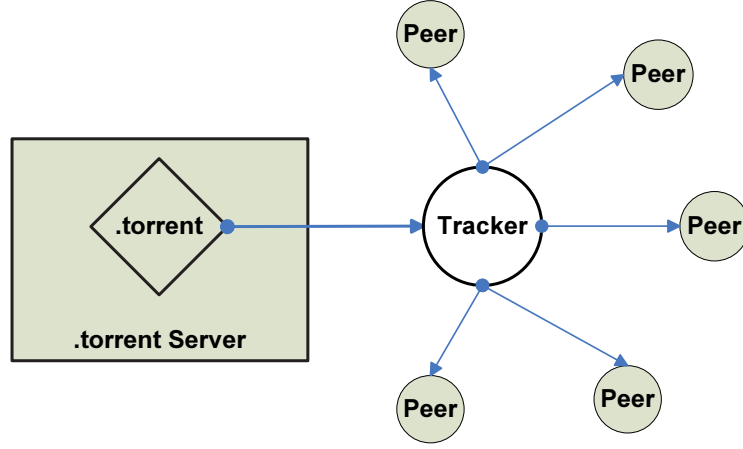


Figure 2.3: Data location service is performed by a centralized server (tracker) in BitTorrent.

its identifier space and routing strategies.

To better understand how DHT-based systems work in reality, we next provide a simple example on the operation of a Chord network [40]. In a Chord network, the key for each node and object are generated using a variant of consistent hashing function [74]. The keys are ordered to form a logical ring. Key k is assigned to the first live network node n whose key is larger than or equal to k . In other words, n holds key k . Note that k and n are all in the same identifier space. Key n is called the successor of key k , denoted by $n = \text{successor}(k)$. Figure 2.4 (derived from [67]) shows a simple example of a Chord network with 10 nodes.

To maintain a consistent hashing mapping when a peer n joins the network, certain keys previously assigned to n 's successor now need to be reassigned to n . When peer n leaves the Chord system, all of its assigned keys are reassigned to n 's successor. Therefore, peers join and leave only require $O(\log^2 N)$ messages to maintain the network structure [40]. No other changes of keys assignment to peers need to occur. To illustrate how a Chord network adapt to peers joining and leaving the network, let's consider the particular ring that has ten peers and stores five keys in Figure 2.4. The prefix K and N before each number represent the identifier (key) is associated with a data object and a peer node, respectively. The successor of the identifier 10 is peer 14, so key 10 will be located at node id 14. Similarly, if a peer were to join with identifier 26, it would store the key with identifier 24 from the

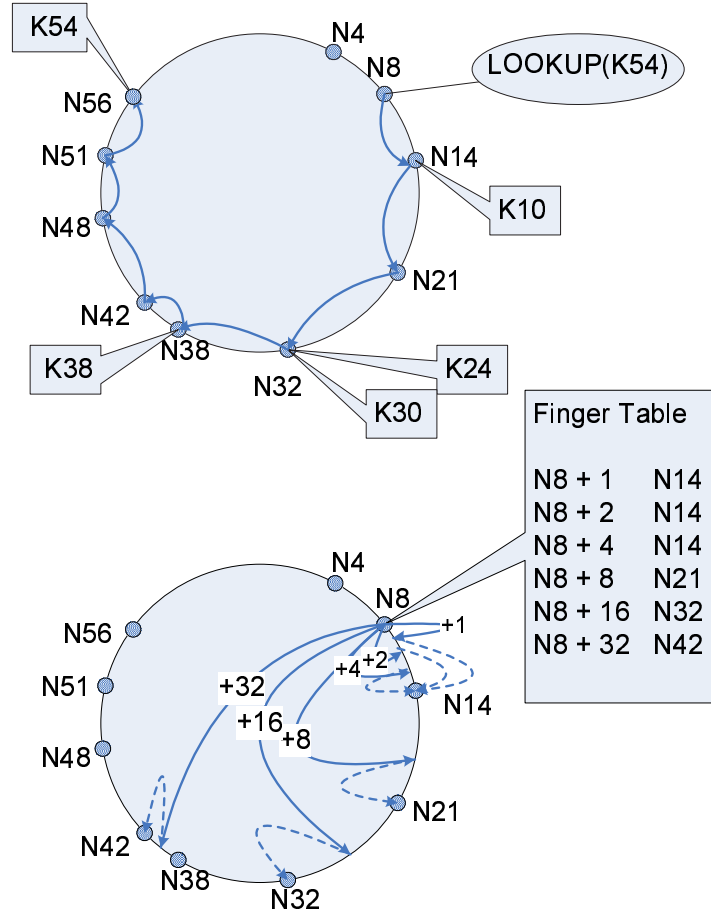


Figure 2.4: Chord ring with identifier circle consisting of ten peers and five data keys. It shows the path followed by a query originated at peer 8 for the lookup of key 54 without the help of Finger table and the Finger table entries for peer 8.

peer with identifier 32.

If each node only has the knowledge about its successor, a query originated from peer 8 for key 54 has to travel 8 hops to reach the destination as shown in the upper part of Figure 2.4. Clearly, relying on immediate successor nodes to relay the lookup messages is very inefficient and unscalable. To make the search length (latency) scalable, each peer implements a finger table, which is a localized routing table, to facilitate data location and message routing. The finger table of a peer node n includes the information of some nodes that is far away from n . Assume the network has 2^m identifiers, the finger table of n

contains the identifiers of nodes associated with identifiers $successor(n + 2^{i-1}), 1 \leq i < m$, so that search messages can be routed to the nodes in the finger table that is more close to the destination. The lower part of Figure 2.4 depicts the finger table entries of peer node 8. The query message originated from peer 8 will traverse through only three peers, which are peers 42, 51 and 56 and hence the search length is greatly reduced. Keeping a localized routing table to facilitate message routing and data location is a very common practice in different types of DHT networks [73, 40, 39, 45].

Although structured P2P network has its advantages over unstructured network, e.g. efficiency in data location and message routing, but it also has drawbacks. First, structured P2P networks have to incur more maintenance traffic to keep the structure when a node join or leave the network. Second, structured peer-to-peer systems is often designed to support exact match lookups while searching with incomplete information, such as keyword searching, can be implemented much easier in unstructured P2P networks. There is no clear winner that suits all application requirements. P2P system designers should choose the network structure that best matches the application requirements.

2.2.2 Data Replication and Peer Selection Strategies

Data replication and peer selection strategies have significant impact on the performance of a P2P network in many aspects. We have already pointed out that peer selection can affect the data location and message routing service of a P2P network. Recall that it is widely known that flooding or random walk searching perform better over a power-law graph. There are some studies that try to form a P2P network into a power-law graph by variants of preferential-attachment [75] to improve the performance of data location service as shown in [69, 46, 70, 71, 72]. In fact, data replication and peer selection strategies affect the application performance of a P2P network in much greater extent. We provide a brief review of the uses and the impact of data replication and peer selection strategies in this section.

In file-sharing applications, peer selection strategies can facilitate the speed at which the file propagates through the network. They do so by utilizing network diversity. Recall from the previous section that KaZaa network already try to utilize the this diversity in a network by constructing a partially decentralized unstructured network, i.e. assigning more important role to peers who are more “capable” to handle more tasks. In KaZaa, peers

with higher computing power or upload bandwidth (supernodes) act as localized message hub or relay. Ordinary nodes rely on supernodes to relay query messages. The same notion of utilizing diversity of network is also applicable when we consider the performance of file downloads. As an intuitive example, when each peer offers different service capacity and there is only one downloading peer, it should always try to receive data from the peers that offers the highest capacity as suggested by [19, 20, 21]. On the other hand, if there are multiple downloading peers in the network, each downloading peer select only the peers with highest upload capacity may not be the optimal solution. The results in [24] suggest that downloading peers should select their source peers probabilistically. An optimal probability assignment scheme for each possible source peers is proposed in [24]. In general, utilizing the heterogeneity of the network helps peers to download their files faster.

Peer selection strategies also help to maintain network connectivity and availability. Network connectivity and availability is important for streaming applications. Note that a peer can have enjoy a smooth playback of some media file only if the network can feed the data to the peer at some minimum rate. When a downloading peer experience a connection loss due to the departure of a connected peer, the data rate at which it receives from the network may not be able to sustain the playback rate and hence causing stutters in playback. The results in [76, 77, 78] show that the lifetime of the peers in the network plays an important role in determining the network connectivity. In other words, the lifetime of the connected peers also determines the playback continuity in P2P streaming applications. When the connectivity of the network is considered as an important issue, it is suggest that a peer should try to connect to peers with longer lifetime than the one with shorter lifetime [79] in an unstructured network to minimize the probability of experiencing a connection loss. Further, there are some attempts that try to incorporate lifetime based peer selection into the network construction methods for structured networks. For example, [26] try to incorporate peers' lifetime information into the construction of a Chord network to reduce the network maintenance traffic. The authors' of [27] try to incorporate the lifetime information when constructing a P2P multicast tree. The results in [27] show that their lifetime-based peer selection helps the existing P2P streaming protocols to improve playback continuity.

In addition to selecting peers based on their capacity or lifetime, selecting peers based on the content that each peer has is also very important. Selecting peers based on

what they have falls into the category of data replication methods. Note that the modern P2P protocols often transfer files by chunks. A chunk is a small fraction of a larger file. Transferring files by chunks helps the network aggregated capacity to scale up faster because peers can start to contribute to the network as soon as it completes downloading one chunk rather than the entire file [16]. However, if the distribution of the chunks are non-uniform, then the “rare” chunks will dictate the network performance [29]. To help the chunks to distribute in the network more evenly, it is proposed in [29] that peers download the chunks that has least number of copies in their local neighborhood. This algorithm is called “rarest-first” algorithm [29] and measurement results show it does a good job at making distribution of chunks towards uniform distribution in [33].

Of course, our overview on the data replication and peer selection strategies are not exhaustive. There are also other data replication and peer selection algorithms that are designed to suit different application requirements. However, in file-sharing or streaming applications, which are the main applications of P2P networks, peer selection indeed plays an very importance role in determining the system performance. As we mentioned in the Chapter 1, the measurement results on the performance of P2P network does not match that predicted by many existing analytical work, which suggest that there must be some factors that are overlooked in analysis. The goal of this study is to find out the reason and try to improve the performance of existing P2P network designs by developing simple peer selection algorithms so that the real world performance can be more close to what we have in theory.

2.2.3 Reputation and Incentive Mechanisms

In a P2P network, the ideal case is that all peers are willing to contribute their resources to others. In reality, it is seldom the case. Developing incentive mechanisms to encourage peers to contribute to the network and thus maintain the scalability is an active research area. To make an incentive mechanism work, peers who contribute often to the network should be rewarded and free-riders, those consume resources but never contribute to the network, should be punished. To punish a free-rider, a peer should reject service requests issued from the free-rider and accept the service requests issued from a “good” peer, which is a peer that contributes to the network a lot.

Obviously, we need some metric to measure the “goodness” or the quality of a peer.

The quality of a peer can have different meanings based on the context of interest. For file-sharing applications, the quality of a peer might refer to the amount of the reciprocated bandwidth. In P2P network formed on top of an ad-hoc wireless networks, it might refer to packet forwarding behavior, e.g. the ratio between the number of packets forwarded and the number of forward requests. With well defined quality metrics, we can build incentive mechanisms on top of two major types of models, namely, reputation and currency models.

The major difference of the two model is the “memory” property. By definition, reputation is an estimate about a peer’s actual quality. In general, reputation models have at least the characteristics listed below [80].

1. Reputation is a function of past behavior and time. A Reputation system should keep track of past behavior.
2. The reputation of a specific peer can be observed either directly and indirectly.

A few examples of reputation systems and their design challenges are presented in [81, 82, 83, 84, 85, 86, 80]. On the other hand, currency-based models often account a single transactions made between peers. Under the currency model, each peer has very short or no memory about the action or behavior of its neighboring peers over time. Each peer’s reaction is an immediate response to its neighbors actions. One example of the currency model is the tit-for-tat algorithm used BitTorrent [29]. The currency in BitTorrent is the bandwidth contributed within one transaction round. Each peer decides whether to reciprocate its bandwidth to its neighboring peers based on the bandwidth (currency) it receives from its neighboring peers in the previous transaction round. Further, in contrast to the reputation-based systems, currency-based system does not rely on second hand observations. There are various incentive algorithms developed under currency-based model [87, 88, 32, 36, 37].

Chapter 3

Uniform Peer Selection Strategies in a Stochastic Environment

3.1 Introduction and Background

In this chapter we briefly describe the characteristics of the service capacity received from the network from the users' perspective. Specifically, we consider the heterogeneity of service capacities over different network paths and the stochastic fluctuation of the capacity over time for a given source peer. In Section 3.2, we show these two important factors are often overlooked in the file download time analysis. In Section 3.3, we show that a simple and distributed algorithm can virtually eliminates all the negative impacts of spatial heterogeneity and temporal correlations, thus greatly reduces the average download time and achieves the simple relation in (1.1) regardless of network settings.

3.1.1 Heterogeneity of Service Capacity

In a P2P network, just like any other network, the service capacities from different source peers are different. There are many reasons for this heterogeneity. On each peer side, physical connection speeds at different peers vary over a wide range [17] (e.g., DSL, Cable, T1, etc). Also, by the type of their access connections, it is reasonable to assume that most peers in a typical P2P network are just personal computers whose processing powers are also widely different. The limitation in the processing power can limit how fast a peer can service others and hence limits the service capacity.

On the network side, peers are geographically located over a large area and each logical connection consists of multiple hops. The distance between two peers and the number of hops surely affect its round-trip-time (RTT), which in turns affects the throughput due to the TCP congestion control. Moreover, in a typical P2P network, this information is usually ‘hidden’ when a user simply gets a list of available source peers that have contents the user is looking for.

Note that the aforementioned factors do not change over the timescale of any typical P2P session (days or a week). Hence, we assume that those factors mainly determine the long-term average of the service capacity over a given source peer.

3.1.2 Correlations in Service Capacity

While the long-term average of the service capacity is mainly governed by topological parameters, the actual service capacity during a typical session is never constant, but always fluctuates over time [89, 90]. There are many factors causing this fluctuation. First, the number of connection a source peer allows is changing over time, which creates a fluctuation in the service capacity for *each user*. Second, some user applications running on a source peer (usually a PC), such as online games, may throttle the CPU and impact the amount of capacity it can offer. Third, temporary congestion at any link in the network can also reduce the service capacity of all users utilizing that link.

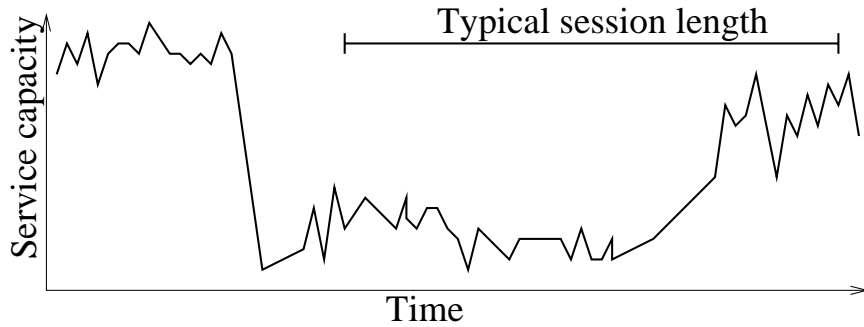


Figure 3.1: Typical variation in end-to-end available bandwidth based on the results in [89, 90]. Drastic changes usually occur in the scale of minutes.

Figure 3.1 shows a typical available end-to-end capacity fluctuation similar to that presented in [91, 89, 90]. The time scale for the figure in the measurement study is on the

order of minutes. We know from [18] that a typical file download session can last from minutes to hours for a file size of several megabytes. This implies that the service capacity over the timescale of one download session is stochastic and correlated. In Figure 3.1, the short-term variations in the capacity are mainly due to the window size fluctuation in TCP, while the long-term variations are due to network congestion, changes in workload or the number of connecting users at the source peer, etc. The long-term fluctuation typically lasts over a longer time scale, say, few minutes up to several hours.

As illustrated in the introduction, both the heterogeneity over different source peers and the correlations of the capacity in a given source peer have significant impact on the average download time. To the best of our knowledge, however, there has been no result available in the literature addressing these issues. All the existing studies have simply assumed that the service capacity is given by a constant (its average value) for the duration of a download. Consequently, the download time of a file of size F is simply given by F/c , where c is the average service capacity. As will be seen later on, however, this is true only when the service capacity is constant or *i.i.d.* over time, neither of them is true in reality. In the next section, we will analyze the impact of these two factors on the per-user performance in terms of the average download time.

3.2 Characterizing the Download Time in a P2P Network

We consider our network as a discrete-time system with each time slot of length Δ . For notational simplicity, throughout this thesis, we will assume that the length of a time slot is normalized to one, i.e., $\Delta = 1$. Let $C(t)$ denote the time-varying service capacity (available end-to-end bandwidth) of a given source peer at time slot t ($t = 1, 2, \dots$) over the duration of a download. Then, the download time T for a file of size F is defined as

$$T = \min \left\{ s > 0 \mid \sum_{t=1}^s C(t) \geq F \right\}. \quad (3.1)$$

Note that T is a stopping time or the *first hitting time* of a process $C(t)$ to a fixed level F .

If $C(t)$, $t = 1, 2, \dots$ are independent and identically distributed (i.i.d.), then by assuming an equality in (3.1), we obtain from Wald's equation [92] that

$$F = \mathbb{E} \left\{ \sum_{t=1}^T C(t) \right\} = \mathbb{E}\{C(t)\} \mathbb{E}\{T\}. \quad (3.2)$$

The expected download time, measured in slots, then becomes $\mathbb{E}\{T\} = F/\mathbb{E}\{C(t)\}$. Note that (3.2) also holds if $C(t)$ is constant (over t). Thus, when the service capacity is *i.i.d.* over time or constant, there exists a direct relationship between the average service capacity and the average download time, as has typically been assumed in the literature.

3.2.1 Impact of Heterogeneity in Service Capacity

We first consider the impact of heterogeneous service capacities of different source peers. In order to decouple the effect of correlations from that of heterogeneity, in this section, we assume that Wald's equation holds true for *each source peer* (i.e., the service capacity of a given source peer is either constant or *i.i.d.* over time). But we allow the average capacities for different source peers to be different. We will consider the impact of correlations in Section 3.2.2.

Let N be the number of source peers in the network (N different end-to-end paths) and $C_i(t)$ be the service capacity of source peer i at time slot t . We assume that $C_i(t)$ is either constant or *i.i.d.* over t such that (3.2) holds. Let $c_i = \mathbb{E}\{C_i(t)\}$ be the average capacity of source peer i . Then, the average service capacity the network offers to a user becomes

$$A(\vec{c}) = \frac{1}{N} \sum_{i=1}^N c_i, \quad (3.3)$$

where $\vec{c} = (c_1, c_2, \dots, c_N)$ and $A(\vec{c})$ is the arithmetic mean of the sequence c_1, c_2, \dots, c_N . Thus, one may expect that the average download time, $\mathbb{E}\{T\}$, of a file of size F would be

$$\mathbb{E}\{T\} = \frac{F}{A(\vec{c})}. \quad (3.4)$$

As we mentioned earlier, however, the actual service capacity of each source peer remains hidden unless a network-wide probe is conducted. So the common strategy for a user is to randomly pick one source peer and keep the connection to it until the download completes. If the user connects to source peer i (with service capacity $C_i(t)$), the average download time over that source peer becomes F/c_i from (3.2). Since the user can choose one of N source peers with equal probability, the actual average download time in this case becomes

$$\mathbb{E}\{T\} = \frac{1}{N} \sum_{i=1}^N \frac{F}{c_i} = \frac{F}{H(\vec{c})}, \quad (3.5)$$

where $H(\vec{c})$ is the harmonic mean of c_1, c_2, \dots, c_N defined by $H(\vec{c}) = [\frac{1}{N} \sum_{i=1}^N \frac{1}{c_i}]^{-1}$. Because $A(\vec{c}) \geq H(\vec{c})$ ¹, it follows that $F/H(\vec{c}) \geq F/A(\vec{c})$. This implies that the actual average download time in a heterogeneous network is always larger than that given by ‘the average capacity of the network’ as in (3.4).

To quantify the difference between (3.5) and (3.4), we adopt similar techniques as in [93]. Let C be the random variable taking values of c_1, c_2, \dots, c_N with equal probability, i.e. $\mathbb{P}\{C = c_i\} = 1/N$ for all i . Consider the following Taylor expansion of the function $f(x) = 1/x$ around some point x_0 :

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2. \quad (3.6)$$

Letting $x = C, x_0 = \mathbb{E}\{C\}$ and taking expectation in both sides of (3.6) give

$$\mathbb{E}\left\{\frac{F}{C}\right\} - \frac{F}{\mathbb{E}\{C\}} \approx \frac{F \cdot \text{Var}\{C\}}{(\mathbb{E}\{C\})^3}. \quad (3.7)$$

From (3.7), we see that the difference between the predicted average download time using (1.1) and the actual average value is governed by two factors, the file size F and the variance of the service capacity, $\text{Var}\{C\}$. First, the actual average download time will be different from (3.4) if the file is large. Second, more importantly, if the service capacities over different source peers vary over a wide range, the actual download time will be much larger than (3.4).

3.2.2 First Hitting Time of a Correlated Stochastic Process

In this section we show that the expected first hitting time of a ‘positively correlated process’ is larger than that of an *i.i.d.* counterpart. Consider a fixed network path between a downloading peer and its corresponding source peer for a file of size F . Let $C(t)$ be a stationary random process denoting the available capacity over that source at time slot t . We will assume that $C(t)$ is positively correlated over time. Then, as before, we can define the download time of a file (or the first hitting time of the process $C(t)$ to reach a level F) as T_{cor} , where the subscript ‘*cor*’ means that $C(t)$ is a correlated stochastic process.

Suppose now that we are able to remove the correlations from $C(t)$. Let $C'(t)$ be the resulting process and T_{ind} be the stopping time for the process $C'(t)$ to reach level F ,

¹The arithmetic mean is always larger than or equal to the harmonic mean, where the equality holds when all c_i ’s are identical.

where the subscript ‘*ind*’ now means that $C'(t)$ is independent over time. Then, again from Wald’s equation, we have

$$\mathbb{E}\{T_{ind}\} = \frac{F}{\mathbb{E}\{C'(t)\}} = \frac{F}{\mathbb{E}\{C(t)\}}.$$

First, as introduced earlier, consider the case that $C(t)$ is 100% correlated over time, i.e., $C(t) = C$ for some random variable C for all t . Then, the download time T_{cor} becomes $T_{cor} = F/C$ assuming an equality in (3.1). Hence, from Jensen’s inequality, we have

$$\mathbb{E}\{T_{cor}\} = F\mathbb{E}\left\{\frac{1}{C}\right\} \geq \frac{F}{\mathbb{E}\{C\}} = \mathbb{E}\{T_{ind}\},$$

i.e., the average first hitting time of an 100% correlated process is always larger than that of an *i.i.d.* counterpart. In order to characterize any degree of positive correlations in $C(n)$, we need the following definition [92]:

Definition 3.1. *Random variables X_1, X_2, \dots, X_n are said to be ‘associated’ if for all increasing functions f and g*

$$\mathbb{E}\left\{f(\vec{X})g(\vec{X})\right\} \geq \mathbb{E}\{f(\vec{X})\}\mathbb{E}\{g(\vec{X})\} \quad (3.8)$$

where $\vec{X} = (X_1, \dots, X_n)$, and we say h is an increasing function if $h(x_1, \dots, x_n) \leq h(y_1, \dots, y_n)$ whenever $x_i \leq y_i$ for $i = 1, \dots, n$. \square

Relation (3.8) characterizes the positive dependence among the random variables X_1, X_2, \dots, X_n . In words, if some of them become larger, then the other random variables are also likely to be larger. Note that (3.8) implies positive correlations in $C(t)$ by setting $f(\vec{X}) = X_i$ and $g(\vec{X}) = X_j$. Definition 3.1 can be generalized to a stochastic process as follows.

Definition 3.2. *The stochastic process $\{X(t), t = 1, 2, \dots\}$ is said to be associated if for all k and t_1, \dots, t_k , the random variables $X(t_1), \dots, X(t_k)$ are associated.* \square

In fact, the set of associated processes comprises a large class of processes. Perhaps, the most popular example is of the following type:

Theorem 4.3.13. in [94]: Let $\{X(t)\}$ be a stochastic process with static space $S = \mathbb{R}^d$ of the form

$$X(t+1) = \varphi(X(t), Z(t)), \quad \text{for } t = 0, 1, \dots \quad (3.9)$$

If the $\{Z(t)\}$ are mutually independent and independent of $X(0)$, then $\{X(t)\}$ is associated if $\varphi(x, z)$ is increasing in x . \square

Stochastic processes of the form (3.9) constitute large portion of Markov processes. For example, any auto-regressive type model with positive correlation coefficient can be written in the form of (3.9). Specifically, for an AR-1 sequence $X(t)$ defined by

$$X(t+1) = \rho X(t) + b\xi(t),$$

where $0 < \rho < 1$ and $\xi(t)$ ($t = 0, 1, \dots$) is a sequence of *i.i.d.* random variables and independent of $X(0)$, we can write $X(t+1) = \varphi(X(t), \xi(t))$ where $\varphi(x, \xi) = \rho x + b\xi$. Since φ is increasing in x , the process $\{X(t)\}$ is associated.

We now present our theorem.

Theorem 3.1. *Suppose that $\{C(t), t \geq 1\}$ is associated. Then, we have*

$$\mathbb{E}\{T_{cor}\} \geq \mathbb{E}\{T_{ind}\}.$$

Proof. First, for any given k , we set $f(\vec{C}) = C(k)$ and $g(\vec{C}) = 1_{\{C(1)+\dots+C(k)>F\}}$, where $\vec{C} = (C(1), C(2), \dots, C(k))$. Note that both functions f and g are increasing. Observe that

$$\{T_{cor} < k\} \equiv \{C(1) + \dots + C(k) > F\}. \quad (3.10)$$

Thus, we have, for any k ,

$$\begin{aligned} \mathbb{E}\{C(k)1_{\{T_{cor} < k\}}\} &= \mathbb{E}\{C(k)1_{\{C(1)+\dots+C(k)>F\}}\} \\ &= \mathbb{E}\{f(\vec{C})g(\vec{C})\} \geq \mathbb{E}\{f(\vec{C})\}\mathbb{E}\{g(\vec{C})\} \end{aligned} \quad (3.11)$$

$$\begin{aligned} &= \mathbb{E}\{C(k)\}\mathbb{P}\{C(1) + \dots + C(k) > F\} \\ &= \mathbb{E}\{C\}\mathbb{P}\{T_{cor} < k\}, \end{aligned} \quad (3.12)$$

where the inequality in (3.11) follows since $C(t)$ is associated, and (3.12) is from the stationarity of $C(k)$ in k and (3.10). Since $\mathbb{E}\{C(k)\} = \mathbb{E}\{C(k)1_{\{T_{cor} < k\}}\} + \mathbb{E}\{C(k)1_{\{T_{cor} \geq k\}}\}$, it follows that

$$\mathbb{E}\{C(k)1_{\{T_{cor} \geq k\}}\} \leq \mathbb{E}\{C\}\mathbb{P}\{T_{cor} \geq k\}. \quad (3.13)$$

Now, let us assume that an equality holds in the definition of T_{cor} (see (3.1)).

Then, we have

$$F = \mathbb{E}\left\{\sum_{k=1}^{T_{cor}} C(k)\right\} = \mathbb{E}\left\{\sum_{k=1}^{\infty} C(k)1_{\{T_{cor} \geq k\}}\right\} = \sum_{k=1}^{\infty} \mathbb{E}\{C(k)1_{\{T_{cor} \geq k\}}\}. \quad (3.14)$$

Substituting (3.13) into (3.14) yields

$$F \leq \sum_{k=1}^{\infty} \mathbb{E}\{C\} \mathbb{P}\{T_{cor} \geq k\} = \mathbb{E}\{C\} \sum_{k=1}^{\infty} \mathbb{P}\{T_{cor} \geq k\} = \mathbb{E}\{C\} \mathbb{E}\{T_{cor}\}.$$

Thus, we have

$$\mathbb{E}\{T_{cor}\} \geq \frac{F}{\mathbb{E}\{C\}} = \mathbb{E}\{T_{ind}\}.$$

This completes the proof. \square

Theorem 3.1 states that the average download time of a file from a source peer with correlated service capacity (in the sense of association defined in (3.8)) is always larger than that of an *i.i.d.* counterpart. In the subsequent section, we show the relationship between the degree of correlation of a process and the average first fitting time of that process, and illustrate how much $\mathbb{E}\{T_{cor}\}$ can be larger than $\mathbb{E}\{T_{ind}\}$. From previous discussions, we know that in general the average download time, $\mathbb{E}[T]$, should be calculated using $\mathbb{E}[F/C(t)]$ rather than the commonly used $F/\mathbb{E}[C(t)]$.

3.2.3 First Hitting Time and Degree of Correlation

To illustrate the relationship between the average download time and the degree of correlation in the available bandwidth $C(n)$, assume that $C(t)$ is given by a stationary first-order autoregressive process (AR-1), i.e.,

$$C(t+1) = \rho \cdot C(t) + \epsilon(t) + \alpha. \quad (3.15)$$

Here, $\epsilon(t)$ is a sequence of *i.i.d.* random variables with zero mean, which represents a noise term of the process. Then, from the stationarity of the process, we get

$$\mathbb{E}\{C(t)\} = \mu = \alpha/(1 - \rho). \quad (3.16)$$

We vary the constant α such that the average capacity is always fixed to $\mathbb{E}\{C(t)\} = \mu = 10$ under different ρ . Since the available bandwidth cannot be negative, we limit the range of $C(t)$ such that $C(t) \in [0, 20]$, while keeping the same mean. The file size is $F = 250$ and the noise term, $\epsilon(t)$, is chosen to be uniformly distributed over $[-1, 1]$, $[-5, 5]$, and $[-9, 9]$ to see how the noise term affects the average download time.

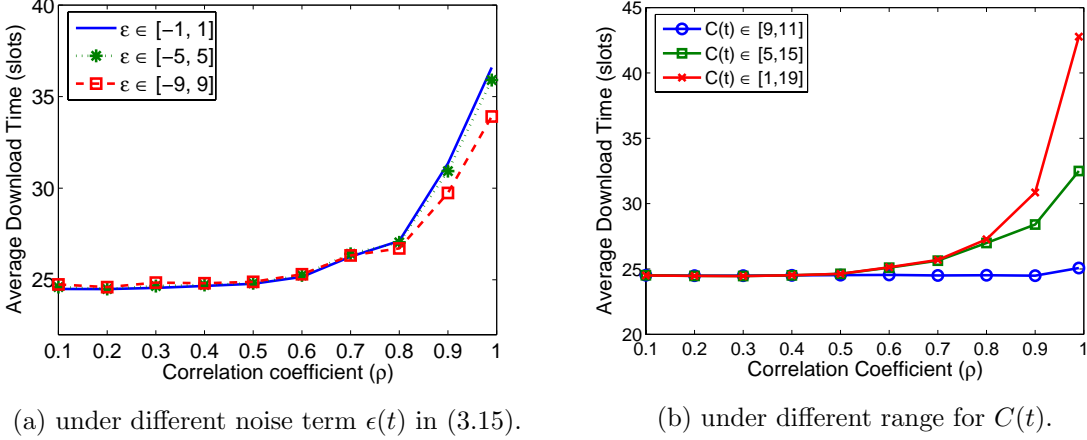


Figure 3.2: Relationship between the average download time and different degrees of correlation ρ .

Remark 3.1. *The choice of the autoregressive process is for the sake of presentation, not to actually reflect the real fluctuation in an end-to-end available bandwidth in real world. It is easy to generate AR-1 process with the same mean but different correlation structures. Similar results can be obtained if the AR-1 process is replaced by other processes with more complicated correlation structures.* \square

Figure 3.2 (a) shows the relationship between the average download time and the degree of correlation of the process (3.15) for different ρ and $\epsilon(t)$. As the degree of correlation increases, the average download time increases. In particular, for a heavily correlated process, the average download time can be about 40% larger than that for an uncorrelated or weakly correlated process, regardless of different noise terms. In other words, the long term variation in the service capacity is the main determining factor of the average download time, and the short-term random noise in the service capacity, such as the one caused by TCP congestion control mechanism over short time scales (RTTs), does not have significant impact on the average download time.

To see the impact of the variance of $C(t)$ itself, we restrict the range of $C(t)$ to some fixed interval. For example, $C(t) \in [9, 11]$ means that we set $C(t) = 9$ whenever it becomes smaller than 9 and $C(t) = 11$ when larger than 11. Figure 3.2 (b) shows the relationship between the average download time and the degree of correlation of $C(t)$ under different variation range for $C(t)$. When the range of fluctuation of $C(t)$ gets smaller, the

download time is less affected by the correlation of the process. This is well expected since the process $C(t)$ fluctuates only within $[9, 11]$ and thus behaves more like a constant process. In contrast, when the range for $C(t)$ is large, the impact of correlation becomes apparent as shown in Figure 3.2(b).

In real data networks, the available capacity of a connection typically shows wild fluctuation; it becomes very low when congestion occurs, and it can reach up to the maximum link bandwidth when things go well. In addition, as technology advances, people are getting links of higher and higher speed, hence the range of available capacity fluctuation is also likely to increase. Therefore, it is very important to consider the effect of correlation in capacity over time when we calculate the average download time of a file transfer.

3.3 Minimizing Average Download Time Over Stochastic Channels

Intuitively, if a downloading peer relies on a single source peer for its entire download, it risks making an unlucky choice of a slow source resulting in a long download. Since the service capacity of each source peer is different and fluctuates over time, utilizing different source peers either simultaneously (parallel downloading) or sequentially within one download session would be a good idea to diversify the risk. Parallel downloading improves the performance by reducing the file size over the ‘worst’ source peer and also may increase the service capacity one receives from the network by utilizing ‘unused’ capacities of other source peers. If a downloading peer utilizes one source peer at a time, switching around seems to be a good strategy to avoid the ‘bad’ source peer. Now, the question is, “What is the criterion for switching, i.e., is it chunk-based or time-based?” In this section we will analyze the performance of (i) parallel downloading, (ii) random chunk-based switching, and (iii) random time-based (periodic) switching.

Different strategies have different impact on the average download time of each peer, which may result in different system dynamics as well, e.g., how fast a downloading peer can start to contribute (become a source peer) or how fast a peer leaves the system after finishing download. If there is no peer leaving the system and all peers are willing to share after they complete their download (either the entire file or a chunk), the aggregate service capacity in the system keeps increasing as time goes on because the number of source

peers continuously grows. In this case, the dynamics in the increase of aggregate service capacity becomes the dominant factor in the average download time for each peer. On the other hand, if no peer is willing to share after download, the aggregate capacity will then eventually drop to zero, thus throttling all the performance metrics. In reality, however, the P2P network will reach a steady-state at some point in which the peer arrivals and departures are balanced and the aggregate service capacity remains around some constant with little variation as shown in [15]. This suggests that the number of source peers in the system will also be around some constant with little fluctuation in the steady-state. In this study, we are mostly interested in the impact of stochastic variations of capacities on the average download time of each peer in the steady-state, rather than in the impact of peer population dynamics in the transient period, which is beyond the scope of this thesis.

Before we start our analysis, we have the following assumptions:

- (i) The service capacity of a source is constant within one time slot.
- (ii) Each downloading peer selects its source independently.
- (iii) Each downloading peer makes blind choice, i.e. the sources are randomly chosen uniformly over all available sources.

Assumption (i) is reasonable since it is expected that there is no major event that triggers dramatic fluctuation in the service capacity within a short period of time. There may be small short-term fluctuations, on the order of seconds, in the service capacity due to the nature of the network protocol, such as TCP congestion window changes, or OS interrupt handling, etc. These changes however do not impose serious impact on the service capacity. Thus, we are not interested in such small short-term variations, but are more interested in the fluctuation on a longer time scale caused by change in the number of connections at a source peer or change in network congestion status, which all usually last for longer time (say, minutes to hours). We have the assumption (ii) because it is impractical for any downloading peer to know how other downloading peers choose their source peers in the network. Hence the downloading peer cannot not make its source selection decision based on other downloading peers' decision. Assumption (iii) is based on the fact that the downloading peer does not know the service capacity of each source peer a priori. Although some protocols require peers to broadcast information about its physical connection speed, it

is hard to tell the “true” instant service capacity of each source peer due to many factors such as competition among other peers, changing workload of the source peer, or the network congestion status. Therefore, a simple way to select a source peer is just to make blind choice.

3.3.1 Random Chunk-based Switching

In the random chunk-based switching scheme, the file of interest is divided into many small chunks just as in the parallel download scheme. A user downloads chunks sequentially one at a time. Whenever a user completes a chunk from its current source peer, the user randomly selects a new source peer and connects to it to retrieve a new chunk. In this way, if the downloading peer is currently stuck with a bad source peer, it will stay there for only the amount of time required for finishing one chunk. The download time for one chunk is independent of that of the previous chunk. Intuitively, switching source peers based on chunk can reduce the correlation in service capacity between chunks and hence reduce the average download time. However, there is another factor that has negative impact on the average download time, the spatial heterogeneity.

First, suppose that there is no temporal correlation in service capacity and Wald’s equation holds for each source peer. A file of size F is divided into m chunks of equal size, and let t_j be the download time for chunk j . Then, the total download time, T_{chunk} , is $T_{chunk} = \sum_{j=1}^m t_j$. Since each chunk randomly chooses one of N source peers (with equal probability), the expected download time will be

$$\mathbb{E}\{T_{chunk}\} = \sum_{j=1}^m \frac{1}{N} \sum_{i=1}^N \frac{F/m}{c_i} = \frac{F}{H(\vec{c})}. \quad (3.17)$$

The result in (3.17) is identical to the download time given in (3.5) where a user downloads the entire file from an initially randomly chosen source peer. In other words, the chunk-based switching is still subject to the ‘curse’ of spatial heterogeneity. While there is no benefit of the chunk-based switching from the average download time point of view, it turns out that this scheme still helps reduce the variance of the download time under a relatively smaller number of users by diversifying the risk with smaller chunks. (See Figure 3.5(b) in Section 3.4.)

In the chunk-based switching, if we get stuck in a source peer with very low service

capacity, downloading a fix amount of bytes from that source peer may still take a long time. We could avoid this long wait by making the size of each chunk very small, but this then would cause too much overhead associated with switching to many source peers and integrating those many chunks into a single file. Therefore, instead of waiting until we finish downloading a fixed amount of data (chunk or file), we may want to get out of that bad source peer after some fixed amount of time. In other words, we randomly switch based on time. In the subsequent section, we will investigate the performance of this random switching based on time and show that it outperforms all the previous schemes in the presence of heterogeneity of service capacities over space and temporal correlations of service capacity of each source peer.

3.3.2 Random Periodic Switching

In this section, we analyze a very simple, distributed algorithm and show that it effectively removes correlations in the capacity fluctuation and the heterogeneity in space, thus greatly reducing the average download time. As the algorithm will be implemented at each downloading peer in a distributed fashion, without loss of generality, we only focus on a single downloading peer throughout this section.

In our model, there are N possible source peers for a fixed downloading peer. Let $C_i(t)$ ($t = 0, 1, 2, \dots$ and $i = 1, 2, \dots, N$) denote the available capacity during time slot t of source peer i . Let $U(t) \in \{1, 2, \dots, N\}$ be a source selection function for the downloading peer. If $U(t) = i$, this indicates that the downloading peer selects path i and the available capacity it receives is $C_i(t)$ during the time slot t . We assume that each $C_i(t)$ is stationary in t and $C_i(t)$ of different source peers $i = 1, 2, \dots, N$ are independent.² We however allow that they have different distributions, i.e., $\mathbb{E}\{C_i(t)\} = c_i$ are different for different i (heterogeneity). For any given i , the available capacity $C_i(t)$ is correlated over time t . As before, when each connection has the same probability of being chosen, the average service capacity of the network is given by $A(\vec{c}) = \frac{1}{N} \sum_{i=1}^N c_i$.

In this setup, we can consider the following two schemes: (i) *permanent connection*, and (ii) *random periodic switching*. For the first case, the source selection function

²We note that different paths (overlay) may share the same link at the network core, but still, the bottleneck is typically at the end of network, e.g., access network type, or CPU workload, etc. Thus, the independence assumption here is reasonable.

does not change in time t . When the searching phase is over and a list of available source peers is given, the downloading peer will choose one of them randomly with equal probability. In other words, $U(t) = U$ where U is a random variable uniformly distributed over $\{1, 2, \dots, N\}$. For example, if the downloading peer chooses u ($u \in \{1, 2, \dots, N\}$) at time 0, then it will stay with that source peer *permanently* ($U(t) = u$) until the download completes.

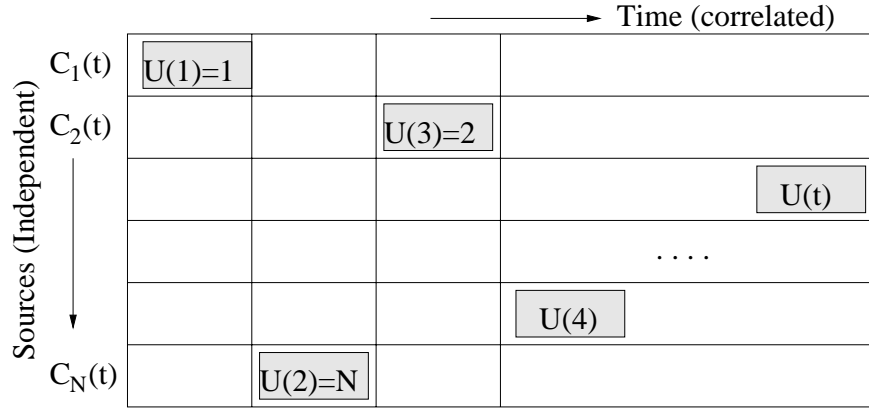


Figure 3.3: The operation of source selection function $U(t)$ for random periodic switching

For the random periodic switching, the downloading peer randomly chooses a source peer at each time slot, independently of everything else. In other words, the source selection function $U(t)$ forms an *i.i.d.* sequence of random variables, each of which is again uniformly distributed over $\{1, 2, \dots, N\}$. Figure 3.3 illustrates the operation of the source selection function $U(t)$ for random periodic switching. In this figure, source 1 is selected at time 1, source N is selected at time 2, and so on.

Let us define an indicator function

$$I_u(t) = \begin{cases} 1, & \text{if } U(t) = u \\ 0, & \text{otherwise.} \end{cases}$$

Then, since $U(t)$ can take values only from $\{1, 2, \dots, N\}$, the actual available capacity at time t can be written as

$$X(t) = C_{U(t)}(t) = \sum_{u=1}^N C_u(t) I_u(t)$$

for both the permanent connection and the random periodic switching strategies. Since each downloading peer chooses a source peer independently of the available capacity, $U(t)$ is also independent from $C_u(t)$, and so is $I_u(t)$. Note that, from $\mathbb{E}\{I_u(t)\} = 1/N$ for any u , we have

$$\begin{aligned}\mathbb{E}\{X(t)\} &= \sum_{u=1}^N \mathbb{E}\{C_u(t)I_u(t)\} \\ &= \sum_{u=1}^N \mathbb{E}\{C_u(t)\}\mathbb{E}\{I_u(t)\} = \sum_{u=1}^N \frac{c_u}{N} = A(\vec{c}),\end{aligned}\tag{3.18}$$

i.e., the average available capacity for the two source selection strategies are the same.

In order to analyze how the two different strategies affect the correlation in $X(t)$, we consider the correlation coefficient of $X(t)$ defined as

$$r(\tau) = \frac{\text{Cov}\{X(t), X(t+\tau)\}}{\text{Var}\{X(t)\}}.$$

Then, we have the following result.

Proposition 3.1. *Let $r_{per}(\tau)$ and $r_{ran}(\tau)$ denote the correlation coefficient of $X(t)$ under the permanent connection and the random periodic switching, respectively. Then, we have*

$$r_{ran}(\tau) = \frac{1}{N} r_{per}(\tau), \quad \forall t \geq 1.$$

□

Proof. Since the average capacity for both strategies remains the same (see (3.18)), without loss of generality, we can assume that $C_u(t)$ for any source peer u has zero mean by subtracting $\mathbb{E}\{C_u(t)\}$ if necessary. From the independence among different source peers, we have, for any $u \neq v$,

$$\mathbb{E}\{C_u(t) \cdot C_v(t')\} = \mathbb{E}\{C_u(t)\}\mathbb{E}\{C_v(t')\} = 0.\tag{3.19}$$

Then, the covariance of $X(t)$ becomes

$$\begin{aligned}\text{Cov}\{X(t), X(t')\} &= \mathbb{E}\left\{\sum_{u=1}^N C_u(t)I_u(t) \cdot \sum_{v=1}^N C_v(t')I_v(t')\right\} \\ &= \mathbb{E}\left\{\sum_{u=1}^N \sum_{v=1}^N C_u(t)C_v(t')I_u(t)I_v(t')\right\} \\ &= \sum_{u=1}^N \sum_{v=1}^N \mathbb{E}\{C_u(t)C_v(t')\}\mathbb{E}\{I_u(t)I_v(t')\}.\end{aligned}\tag{3.20}$$

From (3.19), we can rewrite (3.20) as

$$\sum_{u=1}^N \mathbb{E}\{C_u(t)C_u(t')\}\mathbb{E}\{I_u(t)I_u(t')\}. \quad (3.21)$$

First, consider the case of $t = t'$. Then, it follows that

$$\mathbb{E}\{I_u(t)I_u(t)\} = \mathbb{E}\{I_u(t)\} = \frac{1}{N}.$$

Hence from (3.21) with $t = t'$, the variance of $X(t)$ is given by

$$\text{Var}\{X(t)\} = \frac{1}{N} \sum_{u=1}^N \mathbb{E}\{C_u(t)C_u(t)\} = \frac{1}{N} \sum_{u=1}^N \text{Var}\{C_u(t)\}, \quad (3.22)$$

regardless of the strategies for $U(t)$.

Now, consider the case of $t \neq t'$. Under the permanent connection strategy, since $I_u(t) = I_u(t')$ all the time, we get

$$\mathbb{E}\{I_u(t)I_u(t')\} = \frac{1}{N}.$$

On the other hand, for the random periodic switching, we have

$$\mathbb{E}\{I_u(t)I_u(t')\} = \mathbb{E}\{I_u(t)\}\mathbb{E}\{I_u(t')\} = \frac{1}{N^2},$$

since $I_u(t)$ and $I_u(t')$ for $t \neq t'$ are independent.

Finally, set $t' = t + \tau$. Then, from (3.21) and since the variance of $X(t)$ remains the same for both strategies as in (3.22), we have $r_{ran}(\tau) = r_{per}(\tau)/N$ and this completes the proof. \square

From Proposition 3.1, we see that under the random periodic switching strategy, the correlation of $X(t)$ is N times smaller than that of permanent connection strategy. For example, when each downloading peer has about 10 available source peers ($N = 10$), the correlation coefficient of the newly obtained capacity process under our random periodic switching is no more than 0.1 regardless of the correlations present in the original capacity fluctuation. So, by using our random periodic switching, we can always make the capacity process *very lightly correlated*, or almost independent. From Figure 3.2, we see that the average download time for a lightly correlated process is very close to that given by Wald's

equation. It is thus reasonable to assume that Wald's equation holds for the lightly correlated process $X(t)$ under our random periodic switching strategy. Specifically, if we define T_{ran} as the download time for a file of size F under the random periodic switching, we have

$$\begin{aligned} F &= \mathbb{E} \left\{ \sum_{t=1}^{T_{ran}} C_{U(t)}(t) \right\} = \mathbb{E}\{T_{ran}\} \mathbb{E}\{C_{U(t)}(t)\} = \mathbb{E}\{T_{ran}\} \mathbb{E} \left\{ \mathbb{E} \{C_{U(t)}(t) \mid U(t)\} \right\} \\ &= \mathbb{E}\{T_{ran}\} \frac{1}{N} \sum_{u=1}^N \mathbb{E}\{C_u(t)\} = \mathbb{E}\{T_{ran}\} \frac{1}{N} \sum_{u=1}^N c_u = \mathbb{E}\{T_{ran}\} A(\vec{c}). \end{aligned} \quad (3.23)$$

We then have the following comparison result between the permanent connection and periodic switching.

Proposition 3.2. *Suppose that the process $C_u(t)$ for each u is associated (i.e., it is correlated over time t). Let T_{per} and T_{ran} be the download time for the permanent connection and for the random periodic switching, respectively. Then, we have*

$$\mathbb{E}\{T_{per}\} \geq \mathbb{E}\{T_{ran}\}.$$

Proof. Assume that the file size is F . Since $C_u(t)$ is associated, from Theorem 3.1, we have

$$\mathbb{E}\{T_{per} \mid U = u\} \geq \frac{F}{\mathbb{E}\{C_U(t) \mid U = u\}}, \quad (3.24)$$

for any given source peer u . Observe now that

$$\mathbb{E}\{T_{per}\} = \mathbb{E}\{\mathbb{E}\{T_{per} \mid U\}\} \geq \mathbb{E} \left\{ \frac{F}{\mathbb{E}\{C_U(t) \mid U\}} \right\} \quad (3.25)$$

$$\geq \frac{F}{\mathbb{E}\{\mathbb{E}\{C_U(t) \mid U\}\}} = \frac{F}{\mathbb{E}\{C_U(t)\}} \quad (3.26)$$

$$= \frac{F}{A(\vec{c})} = \mathbb{E}\{T_{ran}\}, \quad (3.27)$$

where (3.25) is from (3.24), (3.26) is from Jensen's inequality and the convexity of a function $f(x) = 1/x$ for $x > 0$, and (3.27) is from (3.23). This completes the proof. \square

Proposition 3.2 shows that our random periodic switching strategy will always reduce the average download time compared to the permanent strategy and that the average download time under the random periodic switching is given by $F/(\vec{c})$ (see (3.26)). Note that this was made possible since the random periodic switching removes the negative impact of both the heterogeneity and the correlations. In addition, our algorithm is extremely simple and does not require any information about the system.

3.3.3 Discussion

So far, we have analyzed the performance of three different schemes that utilize the spatial diversity of the network to improve per-user performance in terms of the average download time. We have considered (i) parallel downloading, (ii) random chunk-based switching, and (iii) random periodic switching. The parallel downloading may perform well if the capacity of each possible source peer is known so as to allocate larger chunks to faster connections and smaller chunks to slower connections. But this method is not practical as one cannot know a priori the service capacity of all source peers. In addition, the service capacity is stochastically fluctuating all the time, and our analysis show that the performance of parallel downloading depends much upon the heterogeneity of the service capacities in different source peers if the chunks are equal in size.

Many P2P applications nowadays use chunk-based file transfer with equal chunk size. As mentioned earlier, the benefit of chunk-based switching is to speed up the conversion from downloading peers to uploading peers and thus indirectly affect the average download time. But, in terms of reducing the average download time directly, it does not help much. Random chunk-based switching may reduce the correlations in the service capacity, but it still cannot eliminate the effect of spatial heterogeneity in different source peers.

In current practice, the chunk based transfer and the parallel download are often combined. Taking BitTorrent and Overnet for examples, a file is first divided into 256KB and 9.5MB chunks of equal size, respectively, and then different chunks are downloaded from different source peers simultaneously. However, we separate the analysis of the two strategies to show how each is different in combating spatial heterogeneity and temporal correlations. Please note that we are not trying to compare the performance of parallel downloading with chunk based transfer since they can be easily combined to yield better performance. Rather, we are comparing the performance of the two strategies with our random periodic scheme. Further, we will present the performance comparison of the combined strategy with the random periodic scheme in Section 3.4.2.

The idea of time-based switching scheme is in fact not new. Such strategy has been implemented in BitTorrent [1] but with some other purpose in mind. In BitTorrent application, by using its optimistic choking/unchoking algorithm, a peer changes one of its servicing neighbors with the lowest upload capacity every 10 seconds in hope to find some

peers offering higher service capacity. However, the idea of switching source peer periodically in the BitTorrent’s optimistic choking/unchoking algorithm is to discover new potential sources rather than to explicitly remove the negative impact of temporal correlations and spatial heterogeneity in service capacity. To the best of our knowledge, we are the first to point out that the random periodic switching gives us the average download time of $F/A(\vec{c})$, while all the other schemes considered so far yield larger average download time.

Our study leads us to believe that the random switching decision should be based on time rather than ‘bytes’ because we are interested in the download time, not the average capacity itself. Indeed, any algorithm based on bytes or a fixed amount of data will suffer the *curse of bad source peer* in that it has to wait until that amount of data is completely received from the ‘bad’ source peer. On the other hand, when the decision is based on time, we don’t need to wait that long as we can jump out of that source peer after a fixed amount of time (one period).

3.4 Numerical Results

In this section we provide numerical results to support our analysis and compare the performance of different schemes for file download under various network configurations. In any case, in our configuration, the fluctuation in service capacity of each source peer is correlated in time. We start with the scenario in which there is only a single downloading peer in the network. We also consider the performance of a P2P network in which there are multiple downloading peers to see how well our proposed algorithm work under a setting that is more close to real world network..

3.4.1 Single Downloading Peer with Heterogeneous Service Capacities

We first show the impact of both heterogeneity and correlations in service capacities on the average download time when there is a single user (downloading peer) in the network. There are $N = 4$ source peers in the network, each offering different average service capacities. Let c_i be the average service capacity of source peer i and $\vec{c} = (c_1, c_2, c_3, c_4)$. The average service capacity of the whole network is then $A(\vec{c}) = (c_1 + c_2 + c_3 + c_4)/4$. We change the heterogeneity in service capacity by changing each c_i , while keeping $A(\vec{c}) = 200kpbs$ the same. We measure the degree of heterogeneity in term of $\delta = \sqrt{\text{Var}\{\vec{c}\}}/A(\vec{c})$, the nor-

malized standard deviation. Table 3.1 shows the different settings used in our simulation in this subsection.

Table 3.1: Average service capacity of each source peer under different configurations.

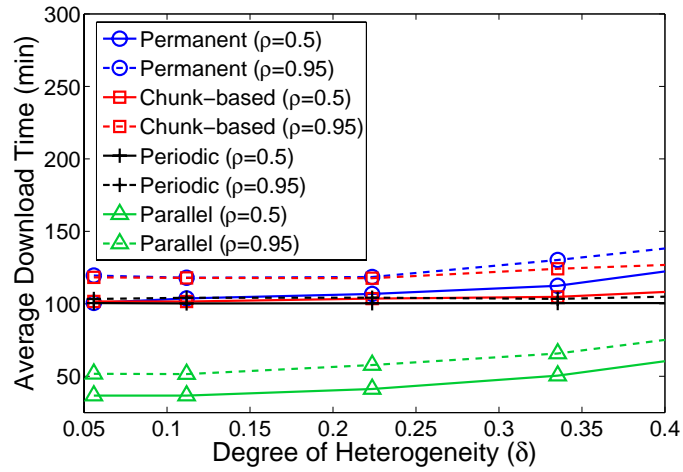
	1	2	3	4	5	6	7	8
c_1	185	170	140	110	80	50	35	20
c_2	195	190	180	170	160	150	145	140
c_3	205	210	220	230	240	250	255	260
c_4	215	230	260	290	320	350	365	380
δ	0.05	0.11	0.22	0.33	0.45	0.56	0.61	0.67

To demonstrate the impact of correlation in each fixed source peer, we use a class of AR-1 random processes to model the stochastic fluctuation in the service capacity. It is reasonable to assume that if the average service capacity is large, the service capacity is more likely to fluctuate over a wider range. For instance, for a high-speed source peer (e.g., 1Mbps), the actual service capacity of the end-to-end session may drop down to somewhere around 50kbps and stays there for a while due to network congestion or limited CPU resources at the source peer. In this regard, we assume that the amount of fluctuation in $C_i(t)$ is proportional to its mean value c_i . Specifically, for source peer i , we set $\epsilon_i(t)$ in (3.15) to be uniformly distributed over $[c_i - \theta_i, c_i + \theta_i]$ where θ_i is chosen such that $\sqrt{\text{Var}\{C_i(t)\}}/\mathbb{E}\{C_i(t)\}$ remains the same for all i .

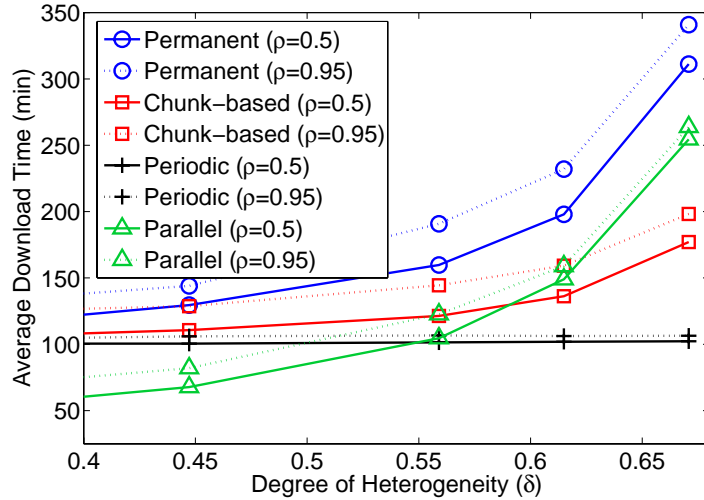
In our simulation, the length of each time slot (one period) is chosen to be 5 minutes. We set the file size to 150MB, which is the typical size of some small video clips or multimedia files. As the average service capacity (of the network) is 200kbps, we set the chunk-size for chunk-based switching to be 7.5MB ($= 200\text{kbps} \times 5 \text{ minutes}$). The purpose of simulating the chunk-based switching is to show the impact of switching based on “data size”, hence we choose 7.5MB to allow fair comparison with the random periodic switching with 5 minute period. We will show the performance of using smaller chunk size later in Section 3.4.2.

We consider all three download strategies discussed so far in comparison with permanent connection. For permanent connection, the user initially chooses one of four sources randomly and stays there until the download completes. For chunk-based switching, the user switches to a new randomly selected source peer whenever a chunk is completed.

Although we simulate the system as a discrete time system, the user is allowed to switch to a new source peer anytime within a time slot whenever it finishes the current chunk. For parallel download, the file is divided into 4 equal-sized pieces and the downloading peer connects to all 4 source peer and download each piece from each source peer simultaneously. Finally, for periodic switching, a user switches to a new randomly chosen source peer every 5 minute to further download the remaining parts of the file.



(a) Low degree of heterogeneity: $0 < \delta < 0.4$



(b) High degree of heterogeneity: $0.4 < \delta < 0.7$

Figure 3.4: Average download time vs. degree of heterogeneity under different download strategies and different degree of correlations.

Figures 3.4 (a)–(b) show the average download time vs. the degree of heterogeneity in the average service capacities (δ) when there is a single downloading peer in the network. Dashed lines are for strong correlations ($\rho = 0.95$) and solid lines represent the case of light correlations ($\rho = 0.5$). In Figure 3.4(a), when the degree of heterogeneity is small, all three single-link download strategies (permanent, chunk-based, periodic) under light correlations perform the same. This is well expected since the service capacities of all source peers are almost *i.i.d.* over space and time, so switching doesn't make any difference and the average download time becomes $F/A(\vec{c}) = 150\text{MB}/200\text{kbps} = 100$ minutes, as commonly used in practice. On the other hand, when there exists strong correlations in the service capacity, the download time is longer for all strategies except the periodic switching. For example, when $\delta = 0.1$, the correlation alone can cause more than 20% of increase in the average download time. Thus, when the network is more like homogeneous (i.e., small δ), the temporal correlation in the service capacity of each source peer becomes a major factor that renders the average download time longer. However, the average download time remains the same under the random periodic switching.

Figure 3.4 (a) also shows the performance of parallel downloading. Intuitively, parallel downloading should perform better than single link downloading because (i) it utilizes more than one link at the same time and (ii) if the connection is poor, parallel downloading reduces the amount of data getting through that bad source peer. Since there is only a single user, it utilizes all the service capacity the network can provide ($c_1 + c_2 + c_3 + c_4$). In this case, the average download time should be $150\text{MB}/(c_1 + c_2 + c_3 + c_4) = 150\text{MB}/800\text{kbps} \approx 25$ minutes. We see from Figure 3.4(a) that parallel downloading can actually achieve the performance close to our expectation when the service capacities of different source peers are close to *i.i.d.* Still, parallel downloading is prone to the negative effect of correlations.

As the degree of heterogeneity increases, the average download time sharply increases for all the schemes except the periodic switching. Figure 3.4 (b) shows this when δ is between 0.4 and 0.7 (see Table 3.1). All but periodic switching suffer from the negative effect of heterogeneity. When both heterogeneity and correlation are high ($\delta = 0.65$ and $\rho = 0.95$), permanent connection takes about 350 minutes to complete the download. This time is about 250 minutes, or 4 hours more than using periodic switching!

It is expected that that the performance of parallel downloading degrades fast when

there is a large degree of heterogeneity. It is more likely that one of the parallel connections is ‘poor’ with very small capacity. Thus, even though the size of chunk (37.5MB) is smaller than the whole file (hence reducing the risk of staying with the bad source peer for too long), this is still not as good as the idea of averaging capacities all the time, as used in the periodic switching. We note that temporal correlations still negatively affect in all these three schemes. However, it should be pointed out that the random periodic switching performs the same *regardless of heterogeneity and correlations*, and in fact it outperforms all the other schemes when the network is heterogeneous with a wide range of service capacities as in the current network.

3.4.2 Multiple Downloading Peers with Competition

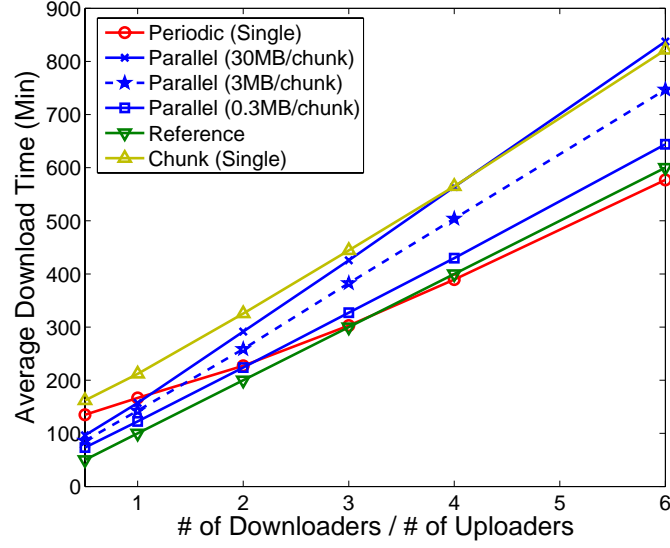
In this section, we consider the performance of different download strategies under a multi-user environment. In our multi-user setting, we set the number of source peers to $N = 100$. The source peers are divided into 4 groups and each source peer within the same group will have the same average service capacity. In reality, the service capacity of each source peer may vary a lot, much greater than the ones that are presented in Table 3.1. We choose the service capacity of the four groups as 1Mbps, 500Kbps, 100Kbps, and 50Kbps, representing typical capacities of LAN, cable, DSL, and modem connections, respectively. In contrast to the setting in the previous section, each group now may consist of different number of source peers to reflect a more realistic distribution of service capacity. We choose the number in each group as 10, 5, 65, 20, respectively. This is to reflect the situation in real world that only few source peers have very high service capacity while most others have the capacity of typical DSL (100Kbps) lines or slower modems. The average service capacity of the network is then $\mathbb{E}\{C\} = 1M \cdot 0.1 + 500K \cdot 0.05 + 100K \cdot 0.65 + 50K \cdot 0.2 = 200\text{Kbps}$. The degree of heterogeneity (δ) in our setting is $\delta = 0.99$. The fluctuation in the service capacity is represented by AR-1 process with correlation coefficient of each source peer set to 0.9. We want to see the performance of different strategies under the impact of spatial heterogeneity and temporal correlation.

In our simulation, service capacity of a source peer is equally divided among all the users connected to that source peer. The effect of dividing capacity among users gives us an idea of how different strategies will perform when users compete for limited resources in the network. To represent the level of competition, we use the *downloading peer to source peer*

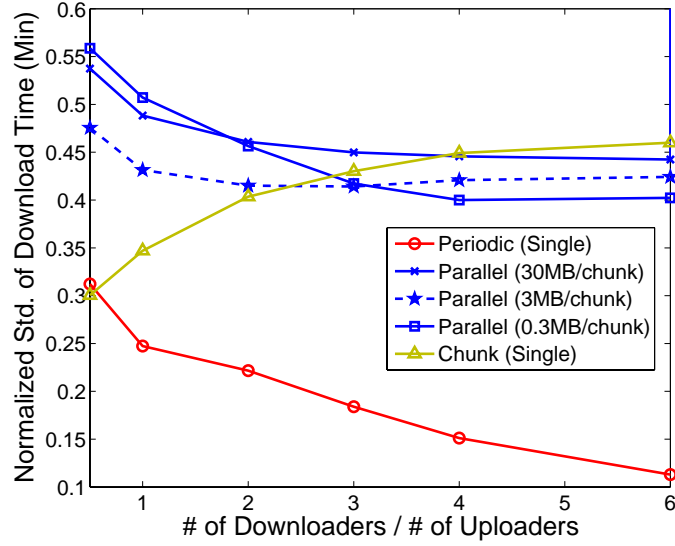
ratio. Since the service capacity of a source peer is equally among the users the source peer serves, we can expect that the service capacity of the system is equally divided among all users as well. Hence, the *average per-user service capacity* can be calculated as the average system service capacity divided by the downloading peer to source peer ratio. For example, if the number of users is 200, then the downloading peer to source peer ratio is 2. The average per-user service capacity will then be $200\text{Kbps}/2 = 100\text{Kbps}$.

We simulate three strategies. First one is the *combined* strategy of parallel download and the chunk-based transfer. Since we know from [56, 52] that keeping only a small number of parallel active connections is better than maintaining connections to all source peers, we set the number of parallel connections to 5 for all the combined strategies. We vary the chunk size to see its impact on the average download time in conjunction with parallel download. Further, the users are allowed to request the same chunk from different source peers when the number of untransferred chunks is less than the number of active parallel connections. For example, if a user is three chunks away from completing the entire file, s/he can request all three chunks from all currently connected source peers. Although making the same chunk requests to different source peers will reduce the download time for that specified chunk, this is at the expense of some waste of the system resource. Note that we do not allow users to make duplicate requests to all connected source peers for every chunk, as this will waste too much resource. This notion of making requests to different source peers for the same chunk when a user's download is nearly complete has been already implemented in BitTorrent called the "end-game" mode [29]. The second strategy is the random chunk-based switching with a single connection. The chunks size is chosen to be 7.5MB, which is identical to what we used in the previous section to allow fair comparison with the periodic switching. Finally, the third strategy is the random periodic switching. The switching period is still 5 minute, but we reduce the length of the system time slot to 1 min. In this case, there will be capacity variations within each switching period.

Figure 3.5(a) shows the average download time for the strategies considered so far. The reference line is given by the file size divided by the average per-user service capacity. First, we can clearly see that the periodic switching performs a lot better than the chunk-based switching. We have a reduction of 40% in average download time by using periodic switching. Next, the combined strategies are shown to outperform the chunk-based switching. Note that when the level of competition is low, the combined strategy



(a) Average download time



(b) Standard deviation / average download time

Figure 3.5: Performance comparison of different strategies under different levels of competition.

outperforms both the chunk-based and the periodic switching schemes. This is well expected because parallelism increases the service capacity each user can achieve in an under-utilized

network. As the level of competition increases, however, the random periodic switching readily starts to outperform the combined strategy. Further, it is interesting to see that the chunk based transfer (7.5 MB per chunk) even outperforms parallel downloading with large chunks (30MB per chunk) when the competition in the system is high. This is because the average download time for parallel downloading is still determined by the slowest link. In a system where there are already many downloading peers, parallelism actually increases the level of competition even more, hence the service capacity of a slow source peer is further divided among its downloading peers.

Another noticeable trend in Figure 3.5(a) is that the performance of the combined strategy gets better with smaller chunk size. Recall that the users can download the same chunk when there are only several chunks left before the completion of the entire file, so the last few chunks will be transferred over the fastest source peer. This method may reduce the negative impact of spatial heterogeneity a little, but at the price of wasting some of the system resource transferring duplicate chunks. The larger the chunk size, the more waste of resources in sending the duplicate chunks. In addition, a larger chunk is more prone to the spatial heterogeneity as the user downloading that larger chunk will have to wait long if it is from a ‘bad’ source. Certainly, very small chunk sizes would make the performance of the combined strategy better and approach the reference line. However, *this comes at a cost*; having small chunks means more protocol overheads because more negotiations between downloading peers and source peers are required. Take the combined strategy using 0.3MB chunks as an example, a downloading peer has to make requests for chunks at least $150\text{MB}/0.3\text{MB} = 500$ times in the entire download session. However, the downloading peer using the periodic switching only needs to make data transfer requests about 120 times in the extreme case (when the downloading peer to source peer ratio is 6). From our simulation result, we can see that random periodic switching is the optimal strategy when the network is over-utilized (the downloading peer to source peer ratio is 3 or higher).

Figure 3.5(b) shows the normalized standard deviation (standard deviation divided by its mean) of the download time for different strategies as the level of competition varies. Larger value of the normalized standard deviation means that the download time among different users will vary more; some can complete the transfer in a very short period while others have to wait for a long time to complete with high probability. Thus, if there is

a large variation in the download time, it is very hard for a user to predict what kind of service s/he will receive. It would be better to have small variations in the download time so that the performance is more predictable and fair. We can clearly see that the periodic switching yields the smallest variation in download time comparing with other strategies we have considered so far.

In summary, the periodic switching not only gives downloading peers the minimal average download time in most network configurations and introduces less overhead, but it is fair with more predictable performance as well.

Chapter 4

Non-Uniform Selection Under Competition and Parallel Connections

Note that the analytical results presented in the previous chapter are based on the assumption that there is only one downloading peer in the network and it uses random uniform peer selection strategy with its single connection. Can we do better than random uniform selection? What should the optimal peer selection strategy be if there are multiple downloading peers competing for the same source peer? In this chapter, we address these questions and provide our answers in stochastic P2P network settings.

4.1 System Model with Multiple Downloading Peers

In a P2P network, all peers can be categorized into two groups for a given file of interest: the peers who are uploading data to other peers (source peers) and those who are downloading from others (downloading peers). We denote the set of source peers and the set of downloading peers by \mathcal{S} and \mathcal{D} , respectively. Each peer can either be in one of the two sets, \mathcal{S} or \mathcal{D} , or in both. For example, a free-rider is in \mathcal{D} only and a peer who contributes to the network without download anything from other peers is in \mathcal{S} only. In most cases, a peer is counted in both sets \mathcal{D} and \mathcal{S} . We use indices i and j to represent the peers in sets \mathcal{D} and \mathcal{S} throughout the thesis, respectively.

We model the network as a discrete time system with the length of a unit time-slot set to Δ . Thus the duration of each slot is $[(t-1)\Delta, t\Delta), t \in \mathbb{N}$. For analytical simplicity, Δ is normalized to 1 and each time slot can then be simply indexed by t . Assume that each downloading peers is able to obtain the information about \mathcal{S} by a well designed search algorithm. In practice, even if \mathcal{S} is known, a downloading peer $i \in \mathcal{D}$ is often *actively connected* to just a subset $\mathcal{S}_i(t) \subseteq \mathcal{S}$ rather than the entire set \mathcal{S} all the time. Note that peers i and j are actively connected if and only if the connection between i and j is carrying data traffic in our definition. For each connection, it is well known that the available bandwidth fluctuates over time [90] due to the workloads of both end points or the network congestion status. Recent studies have shown that most peers in a P2P network utilize broadband connections. Since either cable or DSL lines offer asymmetrical bandwidth and the upstream bandwidth is usually much smaller than downstream bandwidth, it is reasonable to assume that the bottleneck lies in the service capacity of the source peers. Let $C_j(t)$ denote the *total service capacity* a source j can offer at time slot t . We assume that all downloading peers connecting to the same source peer will share the source's service capacity equally, i.e., if there are currently M downloading peers connecting to source j , each downloading peer will get a data rate of $C_j(t)/M$ during time slot t .

Define an indicator function

$$I_{ij}(t) = \begin{cases} 1, & j \in \mathcal{S}_i(t) \\ 0, & \text{otherwise} \end{cases}$$

showing whether downloading peer i connects to source j at time t . Then, $\sum_{i \in \mathcal{D}} I_{ij}(t)$ is the number of downloading peers that are connected to source j at time t . The data rate that the downloading peer i receives from source j at time t becomes

$$R_{ij}(t) = \frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} C_j(t).$$

The aggregated capacity, $R_i(t)$, that a downloading peer i receives from the entire network during time slot t will then be

$$R_i(t) \triangleq \sum_{j \in \mathcal{S}_i(t)} R_{ij}(t) = \sum_{j \in \mathcal{S}} \frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} C_j(t), \quad (4.1)$$

The second equality in (4.1) comes from the fact that $I_{ij}(t) = 0$ for $j \notin \mathcal{S}_i(t)$, i.e., sources not connected to downloading peer i will not contribute to $R_i(t)$.

To simplify the analysis, we impose the following assumptions throughout the thesis.

- (A1) Different downloading peers make their own choices independent of other peers.
- (A2) There is no restriction on the number of connections a source peer can have, i.e., all connection attempts of a downloading peer always succeed.
- (A3) The fluctuation in $C_j(t)$ of a source peer j and the connection decision $I_{ij}(t)$ of a downloading peer i are independent.

We have (A1) because it is unlikely that each downloading peer will broadcast its own peer selection strategy to its neighbors unless the set of downloading peers conspire together. We have (A2) for the sake of simplicity. Without (A2), the analysis is still possible at the cost of much more complicated expressions. Note that we have (A3) because we consider $C_j(t)$, the capacity allocated to P2P applications, be some fraction of the total resource source peer j can offer. Hence, $C_j(t)$ in every time instant will be affected by the workload generated by other non-P2P applications in source peer j or the network congestion status rather than the number of downloading peers that connects to source peer j .

4.2 Download Time Analysis

4.2.1 System Utilization

We first define the system utilization of a P2P network because it is important in determining the average download time. Let the service capacity of source $j \in \mathcal{S}$ be stationary with $\mathbb{E}\{C_j(t)\} = c_j$. We have the following:

Definition 4.1. *The system utilization is*

$$\rho \triangleq \frac{\mathbb{E}\left\{\sum_{i \in \mathcal{D}} R_i(t)\right\}}{\mathbb{E}\left\{\sum_{j \in \mathcal{S}} C_j(t)\right\}} = \frac{\sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{S}} \mathbb{E}\{R_{ij}(t)\}}{\sum_{j \in \mathcal{S}} c_j} \quad (4.2)$$

which is the ratio between the aggregated average service capacity of the entire network and the aggregated average capacity actually consumed by all the downloading peers.

In any network, from an administrator's point of view, the system would perform the best if the "utilization" of the system is maximized. In other words, when the system is under-utilized, i.e. some source capacity being wasted, downloading peers can increase their performance by utilizing those unused resources and thus reduce their download time.

To make our analysis more tractable, let's assume that each downloading peer has the same probability of connecting to a source j , i.e.,

$$\mathbb{P}\{I_{ij}(t) = 1\} = \mathbb{P}\{j \in \mathcal{S}_i(t)\} = p_j, \quad \forall i \in \mathcal{D}. \quad (4.3)$$

Note that p_j ($j \in \mathcal{S}$) is *not* a probability distribution since $\mathbb{E}\{\sum_{j \in \mathcal{S}} I_{ij}(t)\} = \sum_{j \in \mathcal{S}} p_j$ is the average number of connections of downloading peer i , which can be larger than 1 for multiple connections (i.e., parallel downloading). In other words, $I_{ij}(t)$ is *not* independent over j as they are constrained by the number of connections of a downloading peer (usually up to some constant). We note however that $I_{ij}(t)$ is *i.i.d.* over i ($i \in \mathcal{D}$) under (A1) and (4.3).

We now have the following.

Proposition 4.1. *Under (4.3), we have*

$$\rho = \frac{\sum_{j \in \mathcal{S}} [1 - (1 - p_j)^{|\mathcal{D}|}] c_j}{\sum_{j \in \mathcal{S}} c_j}. \quad (4.4)$$

Proof. From definition in (4.2), we have

$$\mathbb{E}\{R_{ij}(t)\} = \mathbb{E}\left\{\frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} C_j(t)\right\} = \mathbb{E}\left\{\frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)}\right\} \mathbb{E}\{C_j(t)\} \quad (4.5)$$

$$= \mathbb{E}\left\{\frac{1}{1 + \sum_{k \neq i, k \in \mathcal{D}} I_{kj}(t)} \middle| I_{ij}(t) = 1\right\} p_j c_j \quad (4.6)$$

$$= \mathbb{E}\left\{\frac{1}{1 + \sum_{k \neq i, k \in \mathcal{D}} I_{kj}(t)}\right\} p_j c_j. \quad (4.7)$$

Note that (4.5) comes from (A3), i.e., $\{I_{ij}(t), \forall i\}$ and $C_j(t)$ are independent. We have (4.6) by using Bayes' rule and $\mathbb{E}\{C_j(t)\} = c_j$. Equation (4.7) comes from the previous observation that, for each j , $\{I_{ij}, i \neq k\}$ and I_{kj} are independent.

Let $X = \sum_{k \neq i, k \in \mathcal{D}} I_{kj}(t)$. Then, since $I_{ij}(t)$ are *i.i.d.* over i , it follows that X is a

binomial random variable with parameter $(|\mathcal{D}| - 1, p_j)$.¹ Then, for $X \sim b(N, p)$, note that

$$\begin{aligned} \mathbb{E} \left\{ \frac{1}{1+X} \right\} &= \sum_{i=0}^N \left(\frac{1}{1+i} \right) \binom{N}{i} p^i (1-p)^{N-i} \\ &= \frac{1}{(1+N)p} \sum_{i=0}^N \binom{N+1}{i+1} p^{i+1} (1-p)^{(N+1)-(i+1)} \\ &= \frac{1}{(1+N)p} (1 - (1-p)^{N+1}). \end{aligned} \quad (4.8)$$

From (4.7) and (4.8), we get

$$\begin{aligned} \mathbb{E}\{R_{ij}(t)\} &= \frac{1}{|\mathcal{D}|p_j} \left[1 - (1-p_j)^{|\mathcal{D}|} \right] p_j c_j \\ &= \frac{1}{|\mathcal{D}|} \left[1 - (1-p_j)^{|\mathcal{D}|} \right] c_j. \end{aligned} \quad (4.9)$$

By substituting (4.9) into (4.2), we obtain (4.4). \square

To better understand the meaning of the system utilization ρ in (4.4), let's consider a simple system with two source peers and two downloading peers where the capacities of two source peers are identical. Suppose that each downloading peer connects to each source with probability $p_j = \mathbb{E}\{I_{ij}\} = 0.5$, but under the constraint that $I_{i1}(t) + I_{i2}(t) = 1, i = 1, 2$, i.e., each downloading peer maintains a connection to exactly one of the two sources at any time. Then, from (4.4), we have $\rho = (1 - (1 - 0.5)^2) = 0.75$. This is indeed true as there are only two possible cases – (i) two downloading peers connect to different sources ($\rho = 1$), or (ii) two downloading peers connect to the same source ($\rho = 0.5$), each of which takes place with the same probability.

From Proposition 4.1, we see that $\rho = 1$ only when $p_j = \mathbb{P}\{I_{ij}(t) = 1\} = 1$ for all j . This suggests that each downloading peer should connect to *all possible source peers* in the network from purely the system utilization point of view. However, in reality, a downloading peer may have many potential source peers and it is impractical to connect to all of them simultaneously, hence we usually have $p_j < 1$.

¹For a binomial random variable X with parameter (N, p) , (we write $X \sim b(N, p)$), we have

$$\mathbb{P}(X = i) = \binom{n}{i} p^i (1-p)^{N-i}, \quad i = 0, 1, \dots, N.$$

4.2.2 Achievable Minimum Average Download Time

Before we show how we can derive the relation between the average download time and the system utilization, we first need a formal definition for the file download time.

Definition 4.2. Let $C(t)$ denote the service capacity that a downloading peer “receives” from the network at time t ($t = 1, 2, \dots$). The file download time T is the first time that the size of the accumulated received data exceeds the file size F . In other words, we have the following equation:

$$T = \min \left\{ T > 0 \mid \sum_{t=1}^T C(t) \geq F \right\} \quad (4.10)$$

The random variable T is the first hitting time of the cumulative process $\sum_{t=1}^T C(t)$ to reach level F . If $\{C(t), t \in \mathbb{N}\}$ are *independent and identically distributed (i.i.d.)*, then by assuming an equality in (4.10), we obtain from Wald’s equation [92] that

$$F = \mathbb{E} \left\{ \sum_{t=1}^T C(t) \right\} = \mathbb{E}\{C(t)\} \mathbb{E}\{T\}. \quad (4.11)$$

The expected download time, measured in slots, then becomes $\mathbb{E}\{T\} = F/\mathbb{E}\{C(t)\}$, which has been widely used in the literature.

However, the equality in (4.11) does not generally hold. Recall that the service capacity of each source peer can be different and can fluctuate over time. First, suppose that a downloading peer is only able to make a *single connection* and waits patiently for its session to complete. This strategy was named “connect-and-wait” strategy in [25]. It was shown in [25] that the temporal correlation in the fluctuation of service capacity can make the average download time over any connection longer. In other words, if a downloading peer selects its source peer by a random variable $J \in [1, \dots, |\mathcal{S}|]$, the average download time T_j , given that $J = j$, usually have the following relationship: $\mathbb{E}\{T_j\} \geq F/\mathbb{E}\{C_j(t)\}$, where equality is achieved only if each of $C_j(t), \forall j \in \mathcal{S}$ is *independent or weakly correlated* over t .

Even if the equality in (4.11) holds for whichever connection the downloading peer chooses, the heterogeneity of the network (different service capacities offered by different source peers) also makes the average download time longer. Suppose now $\mathbb{E}\{T_j\} = F/\mathbb{E}\{C_j(t)\}$ for each j . Then, observe from Jensen’s inequality that $\mathbb{E}_J \left\{ \frac{F}{\mathbb{E}\{C_J(t)|J\}} \right\} \geq \frac{F}{\mathbb{E}_J\{\mathbb{E}\{C_J(t)|J\}\}}$ where the equality is achieved when $\mathbb{E}\{C_j(t)\} = \mathbb{E}\{C_k(t)\}, \forall j, k \in \mathcal{S}$.

Hence, combining the effect of both temporal correlation and network heterogeneity, we arrive to

$$\mathbb{E}\{T\} = \mathbb{E}_J \{\mathbb{E}\{T_J|J\}\} \geq \mathbb{E}_J \left\{ \frac{F}{\mathbb{E}\{C_J(t)|J\}} \right\} \geq \frac{F}{\mathbb{E}_J \{\mathbb{E}\{C_J(t)|J\}\}}, \quad (4.12)$$

where the first inequality comes from the temporal correlation in the fluctuation of the capacity of each source peer and the second inequality comes from the network heterogeneity. Note that (4.12) is for the case in which the network has only one downloading peer and it only utilizes a single connection. We now investigate the performance of the case in which the network has multiple downloading peers in competition, each of which utilizes parallel connections to multiple source peers at the same time.

Assume that each downloading peer i can now have an average of L parallel connections, i.e. $\mathbb{E}\{\sum_{j \in \mathcal{S}} I_{ij}(t)\} = L$, and all downloading peers follow the “connect-and-wait” strategy for each of the connections. We have the following.

Theorem 4.1. *Let $\vec{c} = \{c_j\}$, $j = 1, \dots, |\mathcal{S}|$ represent the vector of average capacities of the source peers. Then*

$$\mathbb{E}\{T_i\} \geq \frac{F}{A(\vec{c})} \frac{\nu}{\rho} \quad (4.13)$$

where

$$\nu = \frac{|\mathcal{D}|}{|\mathcal{S}|} \quad \text{and} \quad A(\vec{c}) \triangleq \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} c_j. \quad (4.14)$$

Note that $\nu/A(\vec{c}) = \left(\sum_{j \in \mathcal{S}} c_j\right)/|\mathcal{D}|$ can be interpreted as the commonly perceived average “share” of system capacity that each downloading receives from the network.

Proof. Assume that a file of size F is divided into $|\mathcal{S}_i|$ pieces because the downloading peer utilizes parallel connections to download from the source set \mathcal{S}_i . Let F_{ij} denotes the size of the piece that the downloading peer i requests from source peer j . Since we are considering the specific downloading peer i , so we suppress the subscript i in F_{ij} . Clearly, we have $\sum_{j \in \mathcal{S}_i} F_j = F$, where $0 \leq F_j \leq F$. Note that p_j is independent of j and let T_{ij} denotes the time required to complete transferring piece F_j from j , as defined in (4.10), and we have

$$\begin{aligned} \mathbb{E}\{T_i\} &= \mathbb{E} \left\{ \max_{j \in \mathcal{S}_i} T_{ij} \right\} = \mathbb{E}_{\mathcal{S}_i} \left\{ \mathbb{E} \left\{ \max_{j \in \mathcal{S}_i} T_{ij} \mid \mathcal{S}_i \right\} \right\} \\ &\geq \mathbb{E}_{\mathcal{S}_i} \left\{ \max_{j \in \mathcal{S}_i} \mathbb{E}\{T_{ij} \mid \mathcal{S}_i\} \right\} = \mathbb{E}_{\mathcal{S}_i} \left\{ \max_{j \in \mathcal{S}_i} \mathbb{E}\{T_{ij}\} \right\}, \end{aligned} \quad (4.15)$$

where $\mathbb{E}_{\mathcal{S}_i}$ is the expectation over the random set \mathcal{S}_i , and the last equality follows from the fact that S_i is randomly generated at $t = 0$, i.e., T_{ij} and S_i are independent.

For each piece of a file that peer i downloads from peer j , we generally have $T_{ij} \geq F_j / \mathbb{E}\{R_{ij}(t)\}$ [25]. Substitute $\mathbb{E}\{R_{ij}(t)\}$ with the result in (4.9) and we have

$$\mathbb{E}\{T_{ij}\} \geq \frac{F_j}{\mathbb{E}\{R_{ij}(t)\}} = |\mathcal{D}| \frac{F_j}{\mu_j c_j},$$

where we let $\mu_j = \left\{1 - (1 - p_j)^{|\mathcal{D}|}\right\}$. Hence, equation (4.15) implies

$$\mathbb{E}\{T_i\} \geq |\mathcal{D}| \mathbb{E}_{\mathcal{S}_i} \left\{ \max_{j \in \mathcal{S}_i} \frac{F_j}{\mu_j c_j} \right\}. \quad (4.16)$$

For any give set \mathcal{S}_i , we know that $F_j = F \cdot \mu_j c_j / \sum_{j \in \mathcal{S}_i} \mu_j c_j$ is a solution to the following optimization problem:

$$\min \left\{ \max_{j \in \mathcal{S}_i} \left\{ \frac{F_j}{\mu_j c_j} \right\} \right\}, \text{ s.t. } \sum_{j \in \mathcal{S}_i} F_j = F, F_j \geq 0.$$

Thus, assume that F_j is allocated proportional to $\mu_j c_j$ and $\sum_{j \in \mathcal{S}_i} F_j = F$, we have

$$\max_{j \in \mathcal{S}_i} \left\{ \frac{F_j}{\mu_j c_j} \right\} = \frac{F}{\sum_{j \in \mathcal{S}_i} \mu_j c_j}. \quad (4.17)$$

Further, note that $I_{ij}(0) = 1_{\{j \in \mathcal{S}_i\}}$ in the “connect-and-wait” strategy for each connection and we have

$$\begin{aligned} \mathbb{E}_{\mathcal{S}_i} \left\{ \sum_{j \in \mathcal{S}_i} \mu_j c_j \right\} &= \mathbb{E} \left\{ \sum_{j \in \mathcal{S}} \mu_j c_j I_{ij}(0) \right\} \\ &= \sum_{j \in \mathcal{S}} \mu_j c_j \mathbb{P}\{I_{ij}(0) = 1\} = \sum_{j \in \mathcal{S}} \mu_j c_j p_j \leq \rho A(\vec{c}) |\mathcal{S}|. \end{aligned} \quad (4.18)$$

Recall from (4.4) that $\rho \sum_{j \in \mathcal{S}} c_j = \sum_{j \in \mathcal{S}} \mu_j c_j$ and $p_j \leq 1$, we have the inequality in (4.18). From (4.16), (4.17) and (4.18), observe that

$$\begin{aligned} \mathbb{E}\{T_i\} &\geq |\mathcal{D}| \mathbb{E}_{\mathcal{S}_i} \left\{ \max_{j \in \mathcal{S}_i} \frac{F_j}{\mu_j c_j} \right\} = |\mathcal{D}| \mathbb{E}_{\mathcal{S}_i} \left\{ \frac{F}{\sum_{j \in \mathcal{S}_i} \mu_j c_j} \right\} \\ &\geq \frac{F |\mathcal{D}|}{\mathbb{E}_{\mathcal{S}_i} \left\{ \sum_{j \in \mathcal{S}_i} \mu_j c_j \right\}} \geq \frac{F |\mathcal{D}|}{\rho A(\vec{c}) |\mathcal{S}|} = \frac{F}{A(\vec{c})} \frac{\nu}{\rho}. \end{aligned} \quad (4.19)$$

This completes the proof. \square

The function $A(\vec{c})$ is *the average system service capacity* from the perspective of a downloading peer. We define the parameter ν as the competition level, which is the average number of downloading peers each source peer has to service. Note that the results in [25] that $\mathbb{E}\{T\} \geq F/A(\vec{c})$ is now just a special case of Theorem 4.1. The schemes described in [25] is the case when $|\mathcal{D}| = L = 1$ and $p_j = 1/|\mathcal{S}|$, which corresponds to $\rho = 1/|\mathcal{S}|$ from (4.4). Therefore, $\nu/\rho = 1$ in (4.13), and we have the result in [25].

From the derivation of Theorem 4.1, we can see that the naive “connect-and-wait” strategy needs to satisfy several conditions in order to achieve the minimum possible average download time, i.e. the equality in (4.13). First, the size for each piece downloaded from different sources must be determined at the beginning of a session such that F_j is proportional to $\mu_j c_j$, i.e $F_j = F \cdot (\mu_j c_j / \sum_{j \in \mathcal{S}_i} \mu_j c_j)$. In reality, it is hard to measure both c_j and μ_j accurately, and hence it is difficult to pre-allocate F_j . Even if we were able to allocate each F_j precisely proportional to $\mu_j c_j$, the possible temporal correlation of fluctuation in the service capacity of each source peer will result in a strict inequality in (4.16) and so does the heterogeneity in the average capacity with different source peers for the inequality in (4.19). The naive “connect-and-wait” strategy for each of the parallel connections is thus unlikely to achieve an equality in (4.13), and therefore we need to consider different algorithms to achieve such equality if at all possible.

4.3 Achieving the Minimum Average Download Time

Theorem 4.1 shows the minimum possible average download time in a stochastic P2P network. In this section, we first show that there exists a way to achieve the equality in (4.13) under all conditions. We then propose a centralized algorithm that maximizes the system utilization ρ , thereby further minimizing the (now achievable) minimum average download time (see (4.13)). Finally, we propose some possible distributed algorithm implementations that may achieve near optimal performance.

4.3.1 Dynamic Peer Selection

As our first step towards developing an algorithm that minimizes the average download time, we show switching peers periodically, i.e. a downloading peer i changes its source set $\mathcal{S}_i(t)$ after each t , can achieve the *equality* in (4.13). As opposed to the “connect-

and-wait” strategy that the set of source peers remains “static” after a session begins, switching peer periodically causes the set of source peer to change dynamically during a download session. We will use the term dynamic peer selection and periodic switching interchangeably in the following sections. Although the notion of periodic switching comes from [25], our setting is much more general. Recall that the network setting in [25] is a *single-user network* in which the only downloading peer i can have a single connection, i.e. $|S_i(t)| = 1, \forall t$ and the downloading peer always select its source peer uniformly at random. Here, we include the effect of using parallel connections, biased peer selection, and the competition among downloading peers in our consideration. In our network setting, the source set $S_i(t)$ of each downloading peer i can change *both in its elements and its size* over time. Each downloading peer can have connection preference for one source peer over another. The capacity of each source peer is shared among the downloading peers connected to it to model the competition.

Under our time-varying dynamic source peer selection scheme, $\mathcal{S}_i(t)$ ($t = 1, 2, \dots$) is a *sequence* of random sets rather than a *constant* set randomly generated at $t = 0$ as the $\mathcal{S}_i(t)$ in Theorem 4.1. The service capacity that a downloading peer i receives from the entire network, $R_i(t)$, then becomes *the sum of random variables over a random set*. Here, we do not enforce a strict limit on the number of parallel connections in *each time slot*. Rather, we assume that the average number is limited to L , i.e. $\mathbb{E}\{\sum_{j \in \mathcal{S}} I_{ij}(t)\} = \sum_{j \in \mathcal{S}} p_j = L$. To show that we can have an equality in (4.13), we need to first show that the temporal correlation in received service capacity of each downloading peer is much reduced by our time-varying dynamic peer selection.

Note that the correlation function of a stationary random process $X(t)$ is defined by

$$\phi_X(\tau) = \frac{\text{Cov}(X(t), X(t + \tau))}{\text{Var}(X(t))}$$

and we have the following.

Theorem 4.2. *For any given downloading peer i , Let $X(t)$ and $Y(t)$ denote the $R_i(t)$ under the “connect-and-wait” and the dynamic peer selection, respectively. In general, we have*

$$\phi_Y(\tau) \leq \max_{j \in \mathcal{S}} \{p_j\} \phi_X(\tau). \quad (4.20)$$

For the special case of uniform (blind) selection, i.e $p_j = L/|\mathcal{S}|$, we have the following

inequality:

$$\phi_Y(\tau) \leq \frac{L}{|\mathcal{S}|} \phi_X(\tau). \quad (4.21)$$

Proof. Let's define a new processes $C'_j(t) = C_j(t) - \mathbb{E}\{C_j(t)\}$, then we have the following from the definition (4.1) that

$$\begin{cases} X(t) = \sum_{j \in \mathcal{S}} \frac{I_{ij}(0)}{\sum_{i \in \mathcal{D}} I_{ij}(0)} (C'_j(t) + c_j) \\ Y(t) = \sum_{j \in \mathcal{S}} \frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} (C'_j(t) + c_j) \end{cases}$$

Note that the processes $X(t)$ and $Y(t)$ are stationary and they both have the same mean and variance. We can compare $\phi_X(\tau)$ and $\phi_Y(\tau)$ by simply comparing $\text{Cov}(X(t), X(t'))$ and $\text{Cov}(Y(t), Y(t'))$, where $t' = t + \tau$. Let's define $\Phi_X(\tau) = \text{Cov}(X(t), X(t'))$ and $\Phi_Y(\tau) = \text{Cov}(Y(t), Y(t'))$ for notational simplicity.

We use the following property of the covariance of random variables:

$$\text{Cov} \left(\sum_j X_j, \sum_j Y_j \right) = \sum_j \sum_k \text{Cov}(X_j, Y_k). \quad (4.22)$$

First, for any given downloading peer i , let

$$A_j(t) := \frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)},$$

and we have the following by applying (4.22)

$$\text{Cov} \left(\sum_{j \in \mathcal{S}} A_j(t) C_j, \sum_{k \in \mathcal{S}} A_k(t') C_k(t') \right) = \sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{S}} \text{Cov}(A_j(t) C_j(t), A_k(t') C_k(t')). \quad (4.23)$$

Note that each single term in (4.23) is

$$\begin{aligned} & \text{Cov}(A_j(t) C_j(t), A_k(t') C_k(t')) \\ &= \text{Cov}(A_j(t) (C'_j(t) + c_j), A_k(t') (C'_k(t') + c_k)) \\ &= \mathbb{E}\{A_j(t) (C'_j(t) + c_j) A_k(t') (C'_k(t') + c_k)\} - \mathbb{E}\{A_j(t) (C'_j(t) + c_j)\} \mathbb{E}\{A_k(t') (C'_k(t') + c_k)\} \\ &= \mathbb{E}\{A_j(t) C'_j(t) A_k(t') C'_k(t')\} - \mathbb{E}\{A_j(t) C'_j(t)\} \mathbb{E}\{A_k(t') C'_k(t')\} \\ &\quad + c_j \mathbb{E}\{C'_k(t')\} (\mathbb{E}\{A_j(t) A_k(t')\} - \mathbb{E}\{A_j(t)\} \mathbb{E}\{A_k(t')\}) \\ &\quad + c_k \mathbb{E}\{C'_j(t)\} (\mathbb{E}\{A_j(t) A_k(t')\} - \mathbb{E}\{A_j(t)\} \mathbb{E}\{A_k(t')\}) \\ &\quad + c_j c_k (\mathbb{E}\{A_j(t) A_k(t')\} - \mathbb{E}\{A_j(t)\} \mathbb{E}\{A_k(t')\}) \\ &= \text{Cov}(A_j(t) C'_j(t), A_k(t') C'_k(t')) + c_j c_k \text{Cov}(A_j(t), A_k(t')) \end{aligned} \quad (4.24)$$

For the process $Y(t)$, the first term in (4.24) can be further simplified into

$$\begin{aligned} & \text{Cov}(A_j(t)C'_j(t), A_k(t')C'_k(t')) \\ &= \mathbb{E}\{A_j(t)C'_j(t)A_k(t')C'_k(t')\} - \mathbb{E}\{A_j(t)C'_j(t)\}\mathbb{E}\{A_k(t')C'_k(t')\} \\ &= \mathbb{E}\{A_j(t)\}\mathbb{E}\{A_k(t')\}\mathbb{E}\{C'_j(t)C'_k(t')\}, \end{aligned}$$

from the independence of $A_j(t)$, $A_k(t')$ and $C_j(t)$. Let $\mathbb{E}\{A_j(t)\} = p_j\mathbb{E}\{1/(1+Z_j(t))\} = p_j\mathbb{E}\{W_j(t)\}$ where $Z_j(t) = \sum_{i \in \mathcal{D}} I_{ij}(t)$. From (A1), the process $Z_j(t)$ is *i.i.d.* and so is $W_j(t)$, we drop the time index t for notational simplicity, i.e. $\mathbb{E}\{W_j\} = \mathbb{E}\{W_j(t)\}$. Recall that $A_j(t)$ and $A_j(t')$ are independent for $t' \neq t$, the second term in (4.24) for $Y(t)$ is zero. Further, $C_j(t)$ and $C_k(t')$ are independent for all $j \neq k$, and (4.23) for the random process $Y(t)$ now becomes

$$\begin{aligned} \Phi_Y(\tau) &= \text{Cov}(Y(t), Y(t')) \\ &= \sum_{j \in \mathcal{S}} \mathbb{E}\{A_j(t)\}\mathbb{E}\{A_j(t')\}\text{Cov}(C'_j(t), C'_j(t')) \\ &= \sum_{j \in \mathcal{S}} p_j^2 (\mathbb{E}\{W_j\})^2 (\mathbb{E}\{C'_j(t)C'_j(t')\}) \end{aligned} \quad (4.25)$$

Now, consider the process $X(t)$, after manipulating the terms, (4.23) becomes

$$\begin{aligned} \Phi_X(\tau) &= \text{Cov}(X(t), X(t')) \\ &= \sum_{j \in \mathcal{S}} \text{Cov}(A_j(0)C'_j(t), A_j(0)C'_j(t')) + \sum_{j \in \mathcal{S}} c_j^2 \text{Cov}(A_j(0), A_j(0)) \\ &= \sum_{j \in \mathcal{S}} p_j \mathbb{E}\{W_j^2\} \mathbb{E}\{C'_j(t)C'_j(t')\} + \sum_{j \in \mathcal{S}} c_j^2 \text{Var}(A_j(0)), \end{aligned} \quad (4.26)$$

where $\text{Var}(X)$ denotes the variance of a random variable X . We can show from (4.26) and (4.25) that

$$\begin{aligned} \frac{\phi_X(\tau)}{\phi_Y(\tau)} &= \frac{\Phi_X(\tau)}{\Phi_Y(\tau)} \\ &= \frac{\sum_{j \in \mathcal{S}} p_j \mathbb{E}\{W_j^2\} \mathbb{E}\{C'_j(t)C'_j(t')\}}{\sum_{j \in \mathcal{S}} p_j^2 (\mathbb{E}\{W_j\})^2 \mathbb{E}\{C'_j(t)C'_j(t')\}} + \frac{\sum_{j \in \mathcal{S}} c_j^2 \text{Var}(A_j(0))}{\sum_{j \in \mathcal{S}} p_j^2 (\mathbb{E}\{W_j\})^2 \mathbb{E}\{C'_j(t)C'_j(t')\}} \end{aligned} \quad (4.27)$$

$$\geq \frac{1}{\max_j \{p_j\}} \quad (4.28)$$

The inequality in (4.28) comes from $\sum_{j \in \mathcal{S}} p_j^2 \leq \max_j \{p_j\} \sum_{j \in \mathcal{S}} p_j$ and $(\mathbb{E}\{W_j^2\})^2 \leq \mathbb{E}\{W_j^2\}$ in the first term of (4.27), and the second term in (4.27) is always non-negative.

Thus we complete the proof. \square

Note that [25] considers only the case of blind selection with $L = 1$. Theorem 4.2 is in a much general form and shows that the temporal fluctuation in *each downloading peer's received capacity* is either uncorrelated or weakly correlated over time so that Wald's equation (4.11) holds true as long as $\max_j \{p_j\}$ is not too large.

Now, suppose that $\max_j \{p_j\}$ is properly selected to be some suitable value, say, 0.5. The correlation in the received capacity from the network is *at least* reduced by 50% compared to “connect-and-wait” strategy from (4.20). Hence, the receive capacity $R_i(t)$ for downloading peer i will be weakly correlated over time. Replacing $C(t)$ in (4.11) directly by $R_i(t)$ in (4.1) gives

$$\begin{aligned} \mathbb{E}\{T_i\} &= \frac{F}{\mathbb{E}\{R_i(t)\}} = \frac{F}{\mathbb{E}\left\{\sum_{j \in \mathcal{S}} \frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} C_j(t)\right\}} \\ &= \frac{F|\mathcal{D}|}{\sum_{j \in \mathcal{S}} \left\{1 - (1 - p_j)^{|\mathcal{D}|}\right\} c_j} \cdot \frac{A(\vec{c})}{A(\vec{c})} \end{aligned} \quad (4.29)$$

$$= \left(\frac{F}{A(\vec{c})}\right) \left(\frac{|\mathcal{D}|}{|\mathcal{S}|}\right) \rho^{-1} = \frac{F}{A(\vec{c})} \frac{\nu}{\rho}. \quad (4.30)$$

Equation (4.29) is a direct application of (4.9) and (4.30) is the result of rearranging the terms in (4.29). Equation (4.30) clearly shows that the equality in (4.13) is achieved by using dynamic peer selection.

We argue that $\max_{j \in \mathcal{S}} \{p_j\}$ is most likely to be strictly less than 1 in practice. If $\max_{j \in \mathcal{S}} \{p_j\} = 1$, it means that the downloading peers will connect to some source peers permanently throughout their entire download sessions. Note that the service capacity of a source peer fluctuates over time. A permanent connection will prevent the downloading peer from switching to some other potential source peers that can offer better capacity at the times when the service capacity of the currently connected source peer plummets.

Further, note that limiting the value of $\max_j \{p_j\}$ also implies limiting the average number of parallel connections a downloading peer can use, i.e. $\sum_{j \in \mathcal{S}} p_j = L \leq \max_{j \in \mathcal{S}} \{p_j\} |\mathcal{S}|$. Although it is a common belief that it is always better to open more parallel connections if a downloading peer wants to complete its download session quicker, such idea is generally not true as will be discussed in Section 4.2. If we are to connect to all possible source peers in the network, we lose the benefit of switching peers (correlation reduction) and hence the average download time will be much larger than the RHS of (4.13). Further, the measurement results in [56, 52] also show that all downloading peers

utilizing parallel connections often do not give better performance than all peers using a single connection. The authors of [56, 52] suggest that the number of parallel connections should be limited. Theorem 4.2 actually agrees with this claim. In the next section, we will investigate the impact of utilizing parallel connections in more detail.

4.3.2 Impact of Parallel Connections

Here, we assume that each downloading peer changes its connections periodically using selecting peers uniformly at random, i.e. $p_j = p = L/|\mathcal{S}|$, $\forall j \in \mathcal{S}$. The average number of parallel connections used by each downloading peer L is chosen to be some small number so that that equation (4.30) holds, and we plot the relation between the average download time $\mathbb{E}\{T\}$, system utilization ρ , and the level of competition ν for different average number of parallel connections L according to (4.30) as Figure 4.1(a) and (b). The network has a fixed number of 40 source peers and the capacity of each source ranges from 1MB/min to 20MB/min with an increment of 0.5MB/min. The number of downloading peers varies from 1 to 200, hence $\nu \in [0.025, 5]$.

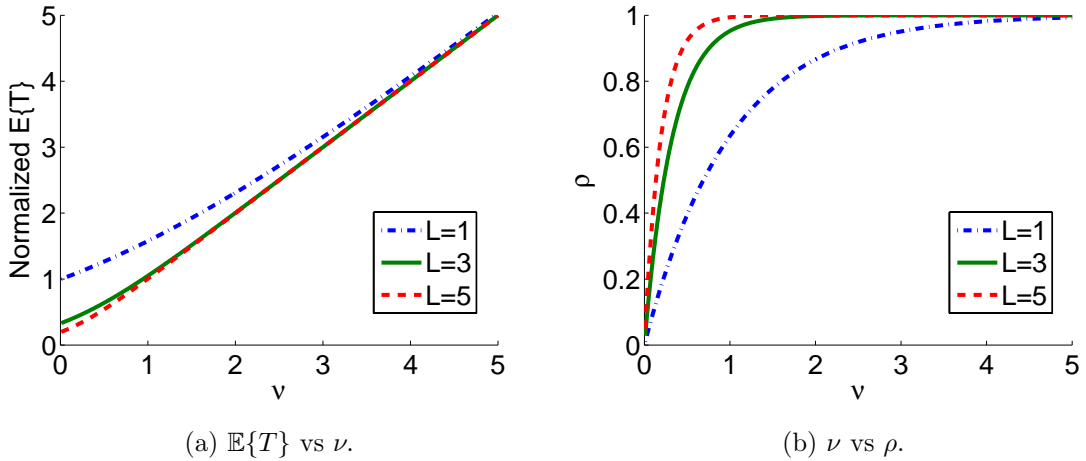


Figure 4.1: The relation between the average download time $\mathbb{E}\{T\}$, system utilization ρ , and the level of competition ν for different average number of parallel connections L . $\mathbb{E}\{T\}$ is normalized over $F/A(\vec{c})$.

By closely examine both Figure 4.1(a) and 4.1(b), we argue that the fundamental effect of parallel downloading is to increase the system utilization rather than directly reducing the average download time. Figure 4.1(a) shows that the benefit of the downloading

peers gain from utilizing parallel connections varies under different network settings. The common belief that parallel connections can help to reduce the average download time is true only when the competition in the network is very low. For example, when $\nu \leq 2$, we see that utilizing parallel connections indeed gives better performance over using a single connection in Figure 4.1(a). The reason is that parallel connections can much increase the system utilization. Taking $\nu = 1$ as an example, Figure 4.1(b) shows that the system utilization is increased from 60% to around 95% by increasing L from 1 to 3. On the other hand, there is no noticeable performance difference between using a single connection and using parallel connections when $\nu \geq 2$. The system performance in this region actually corresponds to the results in [56, 52] that large scale deployment of parallel connections does not always give better performance. Once again, if we refer to Figure 4.1(b), we can see that there is not much room to increase system utilization when ν is large regardless the value of L .

Even when the system is operating at the under-utilized region, utilizing more parallel connections may not guarantee better performance. First, each additional connection does not reduce the average download time by the same amount. Consider the case when $|\mathcal{S}| \gg |\mathcal{D}|L$, and we have the following approximation

$$\rho = 1 - \left(1 - \frac{L}{|\mathcal{S}|}\right)^{|\mathcal{D}|} \approx 1 - \left(1 - L \frac{|\mathcal{D}|}{|\mathcal{S}|}\right) = L\nu.$$

Equation (4.30) approximately becomes $F/(A(\vec{c})L)$ and the average download time is inversely proportional to L . Second, the negative impact of temporal correlation may compromise the benefit we gain from using parallel connections. Consider the case when $\nu \approx 0.5$ in Figure 4.1, we may be able to reduce the average download time a little more by increasing the average number of parallel connections from 5 to 10 or more. However, increasing L implies that we will have less correlation reduction, and it was shown in [25] that different levels of temporal correlation in received capacity could increase the average download time up to 25% or more. Therefore, combining the effect of both parallel connections and temporal correlation, a large L may give the downloading peers worse performance compared with a small L . In summary, the average number of parallel connections should be limited to some small number.

4.3.3 Maximizing System Utilization

From the previous section, we demonstrate that changing the set of source peers dynamically over time can always achieve equality in (4.13) under the condition that both the average number of parallel connections L and the connection probability to each source peer $\max_{j \in \mathcal{S}} \{p_j\}$ are limited. Given that we always have (4.13), what is the minimum average download time? Note that the variables, F , $A(\vec{c})$, and ν in (4.30) are determined once the P2P network is formed. Downloading peers are not able to change these parameters. On the other hand, the system utilization ρ in (4.30) is partially determined by the peer selection probabilities (See (4.4)). Hence, in what follows, we consider how to assign selection probabilities to source peers in order to achieve optimal performance (minimal average download time).

Specifically, we assume that it is possible to measure or obtain the accurate values of $\mathbb{E}\{C_j(T)\} = c_j$ for each of the source peer j in the network. Following the notion in Section 4.2.2, we let each peer have an average of L parallel connections, i.e. $\mathbb{E}\{\sum_{j \in \mathcal{S}} I_{ij}(t)\} = \sum_{j \in \mathcal{S}} p_j = L < |\mathcal{S}|$. From (4.13), minimizing the average download time is equivalent to maximizing the system utilization ρ . From (4.4), we can formulate our problem as follows:

$$\max \quad \frac{\sum_{j \in \mathcal{S}} [1 - (1 - p_j)^{|\mathcal{D}|}] c_j}{\sum_{j \in \mathcal{S}} c_j} \quad (4.31)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{S}} p_j = L, \quad 0 \leq p_j \leq 1 \quad (4.32)$$

Since $(1 - (1 - p_j)^{|\mathcal{D}|})$ is strictly concave in p_j , the problem in (4.31)–(4.32) is a convex optimization problem. We can apply any standard convex programming technique [95] to solve the problem. First, without the loss of generality, assume that the vector $\vec{c} = [c_j, j \in \mathcal{S}]$ is sorted in a decreasing order, i.e., $c_1 \geq c_2 \geq c_3 \dots$. Then, the optimal solution $\{p_j^*\}$ is

$$p_j^* = \begin{cases} 1 - \frac{K^* - L}{\sum_{j=1}^{K^*} \left(\frac{1}{c_j}\right)^{\frac{1}{(|\mathcal{D}|-1)}}} \left(\frac{1}{c_j}\right)^{\frac{1}{(|\mathcal{D}|-1)}}, & j \leq K^* \\ 0, & j > K^* \end{cases} \quad (4.33)$$

Here, K^* is the maximum K such that $\frac{\mu(K)}{|\mathcal{D}|} < \frac{c_K}{C(K)}$, where $C(n) = \sum_{j=1}^n c_j$ and $\mu(n)$ is

defined as

$$\mu(n) = \frac{|\mathcal{D}|}{C(n)} \left[\frac{n-L}{\sum_{j=1}^n \left(\frac{1}{c_j}\right)^{1/(|\mathcal{D}|-1)}} \right]^{(|\mathcal{D}|-1)}. \quad (4.34)$$

For any given $\mathcal{D}, \mathcal{S}, L$, (4.33)–(4.34) gives the optimal peer selection probabilities for any choice of $\{c_j\}, j \in \mathcal{S}$. In a typical P2P network with tens to hundreds of uploading and downloading peers, we argue by an example in the next section that $\max_{j \in \mathcal{S}} \{p_j^*\}$ is most likely to be some small number so that we in general have a significant amount of correlation reduction from Theorem 4.2, and we always have (4.30). Further, $\{p_j^*\}$ maximizes the system utilization ρ in (4.30), and the minimum possible average download time is achieved.

By careful investigation of the expression in (4.33)–(4.34), we can identify the optimal peer selection probabilities in some very special network settings. First, if the network is homogeneous, i.e. all source peers offer the same average service capacity ($c_j = c$ for all $j \in \mathcal{S}$), then $\mu(n)/|\mathcal{D}| = \frac{1}{n} \left(\frac{n-L}{n}\right)^{(|\mathcal{D}|-1)} < 1/n$ for any n . In this case, $K^* = |\mathcal{S}|$ and (4.33) becomes

$$p_j = 1 - \left(\frac{|\mathcal{S}| - L}{|\mathcal{S}| \left(\frac{1}{c}\right)^{1/(|\mathcal{D}|-1)}} \right) \left(\frac{1}{c}\right)^{1/(|\mathcal{D}|-1)} = \frac{L}{|\mathcal{S}|}.$$

Hence, uniform random selection among all possible source peers is the best strategy in a homogeneous environment. Second, recall that $|\mathcal{D}|$ is the number of competing downloading peers in the network. If $|\mathcal{D}|$ is large, i.e. $|\mathcal{D}| \rightarrow \infty$, then we have

$$\lim_{|\mathcal{D}| \rightarrow \infty} \left(\frac{1}{c_j}\right)^{1/(|\mathcal{D}|-1)} = 1$$

in (4.33), regardless of the value of $c_j > 0$. Thus, again, we have $p_j \approx L/|\mathcal{S}|$, i.e., uniform random selection. Note that this uniform random peer selection achieves near-optimal performance *only under very special network configurations* that we just described above, namely, either homogeneous network or a very large number of downloading peers. In all other scenarios such as intermediate number of downloading peers under heterogeneous environment, assigning connection probabilities for each source peer according to (4.33)–(4.34) clearly give far better performance in general. In what follows, we illustrate how the optimal connection probabilities change as the network setting varies.

4.3.4 Optimal Peer Selection Example

Suppose that we have a network with 40 source peers. The source peers are paired into 20 groups. The source peers in each group have the same average service capacity. The average service capacity of the groups ranges from 1MB/min to 20MB/min in increments of 1MB/min. We set the average number of parallel connections $L = 3$ and vary the number of downloading peers $|\mathcal{D}|$ to calculate the optimal vector of connection probabilities using (4.33)–(4.34).

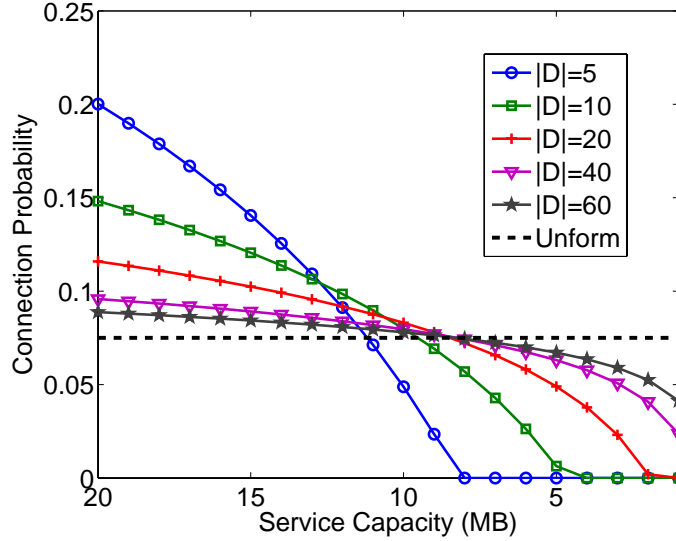


Figure 4.2: The optimal connection probabilities to source peers (from (4.33)) offering different service capacities.

Figure 4.2 shows the computed optimal connection probabilities for the 20 different groups, indexed with their average service capacities from 20MB/min to 1MB/min. All the lines in the figure display a common trend: assigning higher probabilities to source peers offering higher average capacities. Although it is intuitive that a downloading peer should always choose the fastest source peers as [20, 21] suggested, $\max_j \{p_j\}$ is *still very small rather than being very close to 1* even in a very heterogeneous network (we have service capacity of source peers ranging from 1MB/min to 20MB/min) when there is very little competition (e.g. $|\mathcal{D}| = 5$). Therefore, the optimal results based on the analysis of a single-user network does not necessarily apply to the multi-user network settings.

Clearly, Figure 4.2 shows that the probability of connecting to peers offering the average capacity of 20MB/min should increase as $|\mathcal{D}|$ decreases. However, how should the peer selection probability change if the level of competition increases in the network? Figure 4.2 also shows that the optimal connection vector tends to be “flatter” as the number of concurrent downloading peers $|\mathcal{D}|$ gets larger, and it approaches to that of uniform random selection when $|\mathcal{D}|$ goes to infinity, as we have discussed in Section 4.3.3. However, non-uniform peer selection still performs much better than uniform selection even when the level of competition is much larger than 1, i.e. $\nu = |\mathcal{D}|/|\mathcal{S}| > 1$. In other words, a small deviation from uniform distribution, taking the line representing the probability assignment for $|\mathcal{D}| = 60$ as an example, can yield much better performance. We will explicit show the benefit of utilizing non-uniform peer selection by simulation.

Although both the single downloading peer case and infinite downloading peer case are well studied, note that our algorithm finds the optimal vector of connection probabilities between these two extremes under stochastic and heterogeneous environments. Further performance comparison between the optimal peer selection and other selection algorithms will be presented in Section 4.4.

4.3.5 Adaptive Peer Selection Strategy

Note that the centralized peer selection algorithm discussed in the previous section requires the information about the “average” service capacity of each source peer and the number of downloading peers in the network. Such requirement suggests that we need a centralized authority to calculate the optimal connection probability for each downloading peers. Further, broadcasting the optimal solution to all downloading peers introduces much communication overhead. In a system like a P2P network, a distributed algorithm for calculating the connection probability would be more desirable because a distributed algorithm reduces the cost of building a centralized infrastructure and the cost of communication overhead. In this section, we show that we can have a distribution algorithm that achieves near optimal performance with only very little communication overhead.

Our distributed algorithm is a modification of the centralized algorithm in Section 4.3.3. The global parameters in (4.33)–(4.34) are to be replaced by each downloading peer’s own estimates. In other words, each downloading peer estimates the values of c_j and $|\mathcal{D}|$ from its own measurements during its download session. Both c_j and $|\mathcal{D}|$ can be inferred

by the information about the number of downloading peers that are actively connected to a source peer.

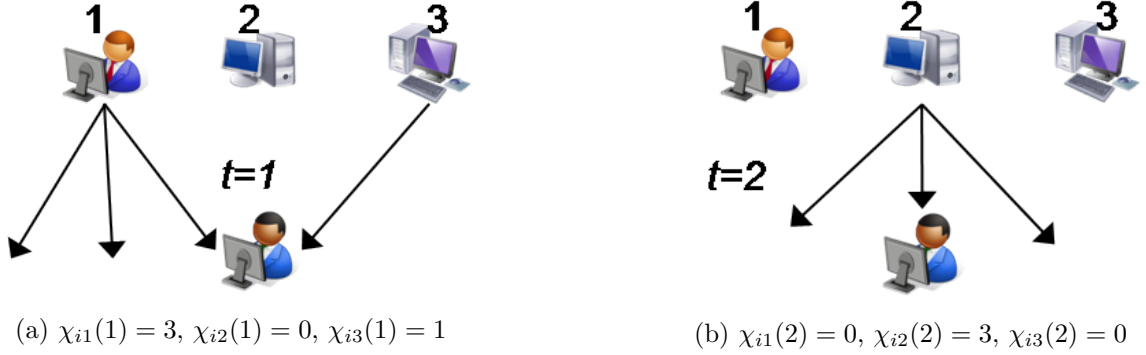


Figure 4.3: A random selected downloading peer i estimates the level of competition in first two time slots.

Let $\chi_j(t) = \sum_{i \in \mathcal{D}} I_{ij}(t)$ denote the number of downloading peer connected to source peer j at time t . Suppose that each downloading peer is now able to obtain the information about $\chi_j(t)$ when it connects to source peer j . We emphasize that the source peers do not broadcast the process $\chi_j(t)$ to all downloading peers in every time slot t . Instead, the downloading peers ask for the value of $\chi_j(t)$ only when they decide to connect to source peer j at t . Such actions introduce very little (if any) communication overhead. Let $\chi_{ij}(t) = \chi_j(t) \cdot I_{ij}(t)$ be the total number of downloading peers sharing the capacity of source peer j , *seen by* the downloading peer i . Figure 4.3 demonstrates a simple example of the values of $\chi_{ij}(t)$ for the first two time slots for a random selected downloading peer i under a network setting in which there are only three potential source peers. In Figure 4.3 (a), the downloading peer i decides to connect to source peers 1 and 3 at the beginning of time slot $t = 1$, respectively. After the connections are established, downloading peer i sees that there are two other peers sharing the capacity of source peer 1, hence, $\chi_{i1}(1) = 3$. On the other hand, downloading peer i is the only peer that is consuming capacity from peer 3, hence $\chi_{i3}(1) = 1$. Since $I_{i2}(1) = 0$, we have $\chi_{i1}(1) = 3, \chi_{i2}(1) = 0, \chi_{i3}(1) = 1$. Similarly, suppose that the downloading peer i decides to connect to source peer 2 at the beginning of the time slot $t = 2$, then we have $\chi_{i1}(2) = 0, \chi_{i2}(2) = 3, \chi_{i3}(2) = 0$ from Figure 4.3.

A downloading peer i can then estimate the average capacity of source peer j by

$$\begin{aligned}\hat{c}_{ij} &= \frac{\sum_{t=1}^T R_{ij}(t) \chi_{ij}(t)}{\sum_{t=1}^T I_{ij}(t)} \\ &= \frac{\sum_{t=1}^T \left(\frac{I_{ij}(t)}{\sum_{i \in \mathcal{D}} I_{ij}(t)} C_j(t) \right) ((\sum_{i \in \mathcal{D}} I_{ij}(t)) I_{ij}(t))}{\sum_{t=1}^T I_{ij}(t)} \\ &= \frac{\sum_{t=1}^T I_{ij}(t) C_j(t)}{\sum_{t=1}^T I_{ij}(t)}\end{aligned}$$

Clearly, \hat{c}_{ij} is an unbiased estimator for $c_j = \mathbb{E}\{C_j(t)\}$.

We argue heuristically that $\chi_{ij}(t)$ can be used for estimating the number of downloading peers in the network as well. Recall that each downloading peer has an average of L parallel connections, hence the average total number of connections is $|\mathcal{D}|L$. Note that $\hat{\chi}_{ij} = \frac{\sum_{t=1}^T \chi_{ij}(t)}{\sum_{t=1}^T I_{ij}(t)}$ is the estimate for $\mathbb{E}\{\sum_{i \in \mathcal{D}} I_{ij}(t)\}$, which is the average number of downloading peers connected to source peer j , from the downloading peer i 's point of view. Summing up the average number of downloading peers connected to source peer j gives the total average number of connections in the network, i.e., $|\mathcal{D}|L$. Thus, each downloading peer i can use $\sum_{j \in \mathcal{S}} \hat{\chi}_{ij}$ to estimate the number of all downloading peers $|\hat{\mathcal{D}}|$ in the network. Then, the downloading peers can use their estimated system parameters to calculate the connection probability to each source peer by replacing c_j and $|\mathcal{D}|$ in (4.33) and (4.34) with \hat{c}_{ij} and $|\hat{\mathcal{D}}|$.

Note that all the calculation so far is based on each peer's own observation from the network. Since no observation can be made without connection, we have to let $p_{ij} \geq \epsilon > 0$, where ϵ is a very small number to ensure that each downloading peer is able to observe and estimate the global information rather than simply ignores some source peers completely. Algorithm 1 summarizes our distributed algorithm for connection probability assignment.

In Algorithm 1, S_{ij} holds the number of times that a downloading peer i connects to a source j . Only two variables $R_{ij}(T)$ and $\chi_{ij}(T)$ are needed from each source j in time slot T to carry out the connection probability assignment. Here, we briefly explain the key steps of Algorithm 1 in each time slot. From line 6 and 7, we first connect to L source peers that have not been connected ($S_{ij} = 0$) at a time to get a global estimates on the parameters for the network as quick as possible. When $S_{ij} > 0$, $\forall j$, a downloading peer i can then calculate the connection probability for each source peer j from lines 9 to 12 using (4.33) and (4.34) with the estimated parameters. Lines 14 – 16 are the steps to update each

Algorithm 1 Distributed Connection Probability Assignment

```

1:  $S_{ij} := \sum_{t=1}^T I_{ij}(t)$  with initial value 0
2: In each time slot  $T$ :
3:  $R_{ij}(T) :=$ received capacity from a source  $j$ .
4:  $\chi_{ij}(T) :=$ observed number of peers connected to  $j$ .
5: while !(file complete) do
6:   if  $\exists S_{ij} = 0$  then
7:     Connect to  $L$  source peer from  $\{j|S_{ij} = 0\}$  uniformly. Let  $G$  denotes the selected
       set.
8:   else
9:      $|\hat{\mathcal{D}}|_i = \frac{\sum_j \hat{\chi}_{ij}}{L}$ 
10:    Calculate  $p_{ij}$  using  $|\hat{\mathcal{D}}|_i, \hat{c}_{ij}$  according to (4.33) and (4.34) with  $L$  in (4.32) replaced
      by  $L' = L - |\mathcal{S}|\epsilon$ 
11:     $p_{ij} = p_{ij} + \epsilon$ 
12:    Connect to source peer  $j$  with probability  $p_{ij}$ . Let  $G$  denotes the set of connected
      source peers.
13:   end if
14:    $S_{ij} := S_{ij} + 1 \ \forall j \in G$ .
15:    $\hat{c}_{ij} := \frac{(S_{ij}-1)\hat{c}_{ij}+R_{ij}(T)\cdot\chi_{ij}(T)}{S_{ij}} \ \forall j \in G$ .
16:    $\hat{\chi}_{ij} := \frac{(S_{ij}-1)\hat{\chi}_{ij}+\chi_{ij}(T)}{S_{ij}} \ \forall j \in G$ .
17: end while

```

downloading peer i 's estimated parameters, namely \hat{c}_{ij} and $\hat{\chi}_{ij}$. We have to emphasize again that the estimates of both \hat{c}_{ij} and $\hat{\chi}_{ij}$ cost almost no communication overhead because they are updated only when i decides to connect to j . In summary, our Algorithm 1 is simple and fully distributed. In Section 4.4, we show the performance of Algorithm 1 via NS-2 simulations.

4.4 Simulations

In this section, we use NS-2 simulations to compare the performance of peer selection strategies in a P2P network. A simple illustration of our network setting is depicted in

Figure 4.4. The Internet cloud consists of many inter-connected nodes including backbone routers and edge routers. We assume that the Internet backbone has high bandwidth and does not introduce any congestion, implying that the main bottleneck is the access link of each peer. Such assumption enables us to run simulations using a network topology similar to a star-shaped topology. We first setup a meshed network of five core routers. The link between any two routers has a bandwidth of 1Gbps with delay of 10ms. We have 50 peers connecting to the core routers and the access links of the peers have different bandwidth limits. To reflect a general network setting for the access bandwidth, we configure 10% of the total 50 peers to have 10Mbps upstream (from peer node to the center node) capacity limit, 20% to have 5Mbps, 40% to have 1Mbps and the rest to have 100Kbps. These groups represent situations in typical LAN, high speed cable/DSL, low speed cable/DSL, and modem connections, respectively. We set 10Mbps as the downstream capacity limit so that the transmission bottleneck will most likely be at the source peers.

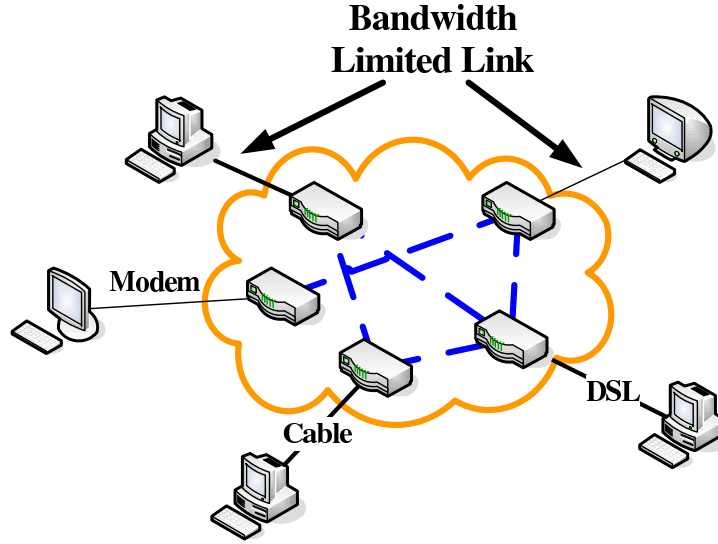


Figure 4.4: The illustration of the network used in our NS-2 simulation. We assume that the Internet backbone does not introduce any congestion. The upstream access link of each peer is bandwidth limited.

Note that each node's upstream bandwidth is not only shared among other peers connected to it but also by some other network applications as in real world case. We model those non-P2P traffic by using the on/off Pareto traffic generators in NS-2. We set both the

average “on” and “off” periods to be 10 minutes. When a traffic source is in the “on” state, it generates a constant bit-rate traffic at 90% of the full upstream capacity. Hence, the long term upstream average capacity for P2P traffic will be 55% of the access link bandwidth limit. The P2P file transfer traffic is carried over the TCP connection in NS-2. Note that in our NS-2 simulations, the actual capacity fluctuation in each of connections of a peer is governed by not only those asymmetric physical bandwidth limits, but also the actual TCP congestion control algorithm, the number of concurrent connections to the source peer, and the random Pareto on/off background traffic, i.e., it is stochastic in time and heterogeneous over space. As we have stated in Chapter 1, we are only interested in the peer selection algorithm itself rather than the problem of free-riders. All peers can serve as source peers, hence $|\mathcal{S}| = 50$. The number of downloading peers ranges from 10 to 50. Here, we set the number of average parallel connections $L = 2$.

First, we show the performance difference between the simple “connect-and-wait” strategy and dynamic peer selection with no connection preference (uniform selection). Under the “connect-and-wait” strategy, the file is divided into 2 pieces of equal size. Under dynamic peer selections, we set the basic data transfer unit (chunk) to be 16KB and each time slot is set to be 1 minute. We choose these two strategies for separate comparison purposes because “periodic uniform peer selection” alone already makes drastic improvement over the “connect-and-wait” strategy. We then compare the performance between periodic uniform peer selection and other algorithms to demonstrate the effectiveness of the optimal strategies more clearly. We first set the file size $F=50\text{MB}$ for the ease of presenting our results. We have also run simulations using larger file sizes and observed the same result. (“connect-and-wait” strategy always performs worst)

Figure 4.5 shows the results for our first scenario. The line marked with “uniform” is the result of the periodic uniform selection, and the line marked with “permanent” is the result for the “connect-and-wait” strategy. In all cases, we can see at least 50% reduction in average download time by using periodic uniform peer selection. In some extreme cases, take $|\mathcal{D}| = 50$ as an example, the average download time for “connect-and-wait” strategy is almost 4 times longer than the periodic uniform selection. The result is exactly what we anticipated from our discussion in Section 4.2. Next, we show that under the dynamic peer selection schemes (non-uniform), the average download time can be further shortened by selecting peers using the optimal connection probabilities. The duration of each time

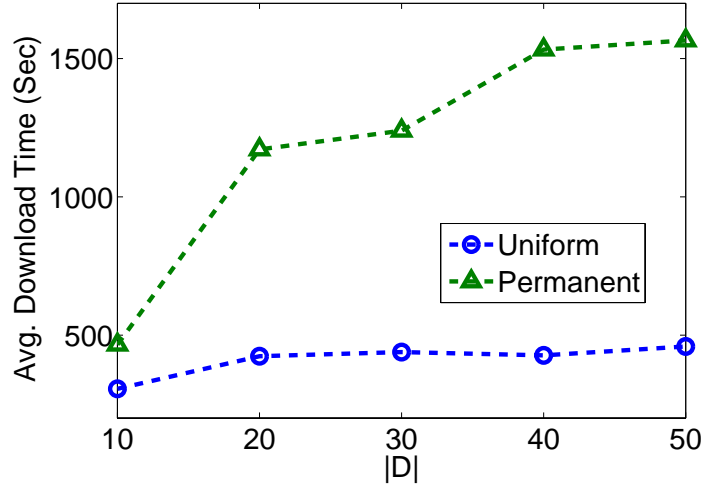


Figure 4.5: The average download time for periodic uniform peer selection and “connect-and-wait” strategy. There are 50 nodes and no free-riders, hence $|\mathcal{S}| = 50$. The number of downloading peers varies from 10 to 50.

slot is still 1 minute. We compare the performance between the uniform strategy (periodic uniform peer selection), the centralized optimal selection strategy in (4.33)–(4.34) and the distributed strategy (Algorithm 1) in Section 4.3.5. We change the file size to 200MB, which is typical for some video clips that contribute to long sessions so as to reflect a more general and realistic case.

Figure 4.6 shows the average download time along with its 95% confidence interval for three different dynamic peer selection strategies. It is clear that both the centralized optimal strategy and the distributed strategy offer a performance boost over the simple uniform peer selection. Note that this simple uniform selection is still much better than the usual connect-and-wait strategy as shown in Figure 4.5. Comparing the performance difference between uniform peer selection and that of our centralized algorithm, we have to emphasize again that even if our numerical example in Section 4.3.4 shows the optimal peer selection probability tends to move towards a uniform distribution when the level of competition increases, a subtle deviation from uniform distribution still gives us significant performance improvements.

It is clear that the optimal strategy can further reduce about 50% of the average download time compared with the periodic uniform selection strategy over most range of

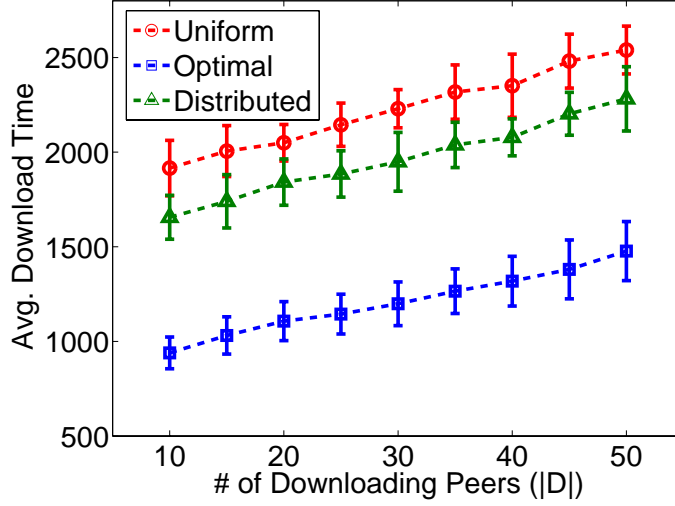


Figure 4.6: The average download time for three dynamic peer selection strategies: periodic uniform peer selection (Uniform), the centralized optimal strategy in (4.33)–(4.34) (Optimal), and the distributed strategy in Algorithm 1 (Distributed).

values of $|\mathcal{D}|$. We see that our simple distributed algorithm derived from the centralized optimal strategy can still reduce the average download time by 10% to 15%. Clearly, there is a performance gap between the distributed algorithm and the centralized optimal strategy. This is inevitable due to the estimation errors in our simple distributed algorithm. To confirm our hypothesis, we redo the simulation using a file with 400MB in size. We plot the ratio between the average download time using non-uniform peer selection and uniform peer selection under different file sizes in Figure 4.7 to demonstrate the effect of estimation error. The value in the Y-axis is the fraction of the average download time of non-uniform selection algorithms over the average download time of uniform peer selection.

As we can see from Figure 4.6, the average download time for the distributed algorithm ranges from 1500 to 2000 seconds, which means that a downloading peer can switch its source peers 25 to 35 times on average. Hence, each peer will make around 60 observations of the system ($L = 2$). Note that we have 50 source peers in the network and such number of observations (60) is not enough to obtain accurate estimates for c_j , especially under highly stochastic and heterogeneous environment as in our setting. In other words, The estimate \hat{c}_{ij} may not be close to the real value of c_j . On the other hand,

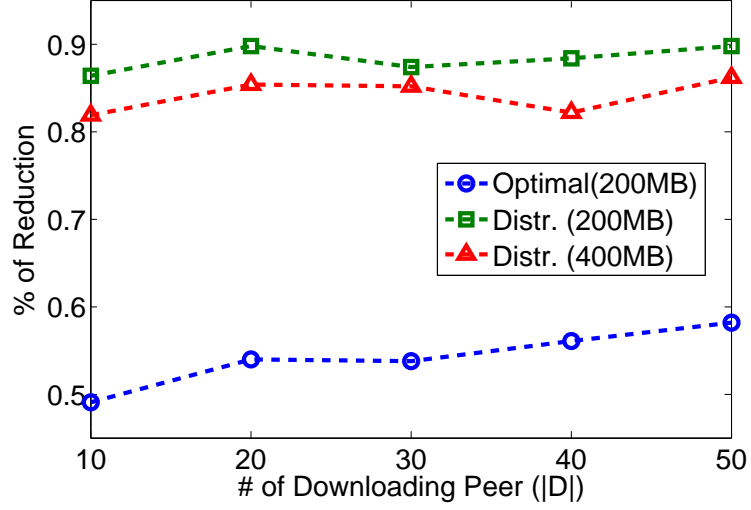


Figure 4.7: The reduction in the average download time for different peer selection strategies over the uniform peer selection.

we have noticeable performance improvement when we increase the size of the file from 200MB to 400MB. Note that the file download session for a 400MB file is certainly longer than that for a 200MB file in absolute value. Hence, each downloading peer is able to obtain a more accurate estimates on the required system information, namely the $|\hat{\mathcal{D}}|_i$ and \hat{c}_{ij} , respectively. Hence, the level of download time reduction is greater when downloading a 400MB file. We can see clearly from Figure 4.7 that the line representing the reduction under 400MB file is more closer to the line representing the result of using centralized algorithm.

It is obvious that we can obtain more accurate estimates about the system by transferring larger files or making each time slot smaller. However, making each time slot small has its drawbacks, i.e. increasing the overhead for connection establishment and tear-down. Further, from our simulation results, we may achieve performance close to optimal if we download a file much larger than 400MB (1GB perhaps). However, not all file downloads involves files with huge size. Hence, our distributed algorithm has its limitations in real world practice in its current form. An immediate extension of our work is to develop some distributed algorithms that can gather and estimate system information with great accuracy so that we can achieve significant download time reduction under small file size while keeping the size of each time slot reasonable.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In this thesis, we jointly consider many factors that have impact on the average download time, which are often considered separately in the literature, and derive an optimal centralized peer selection strategy that greatly improves content delivery performance in a stochastic heterogeneous P2P network. In contrast to the commonly-held practice focusing on the notion of average capacity, we have shown that both the spatial heterogeneity and the temporal correlations in the service capacity can significantly increase the average download time of the users in the network, even when the average capacity of the network remains the same. From our results in Chapter 3, it becomes apparent that all P2P algorithms regarding the download time should focus directly on ‘time’ rather than on ‘bytes’, and the notion of average service capacity alone is not sufficient to describe each user’s average performance in a P2P network.

Further, we address some of the often neglected issues, namely, the competition for resource among multiple concurrent downloading peers, jointly with the spatial heterogeneity and temporal correlation in service capacities of the source peers in Chapter 4. We derive the relationship between the average download time and system performance metrics such as system utilization and level of competition, and develop our optimal peer selection strategies based on this relationship. We are also able to explain the fundamental impact of using parallel connections in such networks. We show that a simple heuristic suboptimal distributed algorithm derived from our centralized one can greatly improve performance

over the uniform random peer selection, with almost no computational overhead.

5.2 Future Work

One of the major characteristics of a P2P network is self-organization. In other words, peers must have significant or total autonomy from central servers, which implies that a good peer selection algorithm should be completely distributed. From the simulation results presented in Chapter 4, our distributed algorithm is suboptimal and there exists a space for performance improvement. Exploring more approaches towards developing a distributed algorithm that can achieve performance closer to that of our centralized peer selection algorithms is an immediate extension of the result of this thesis.

With the rapid growing of streaming applications that utilize P2P technology, we should develop peer selection strategies that are suitable for media streaming applications. Although file-sharing and streaming applications share the same general purpose: spread a file in the network, streaming applications have much more strict constraints. In file-sharing systems, the average download time is often the only metric and hence peers often try to maximize their data rate. On the other hand, in streaming applications, not only the data rate is important but the time and the order at which contents (Bytes) arrive at the receiving peer is also important. Data arrives after the playback deadline or out of order is often considered useless. Useless data causes disruption in stream playback. A future research direction would be incorporate the idea of switching source peers periodically into peer selection algorithms in P2P streaming applications. We identify some of the important issues and challenges for our future research direction as follows.

- 1 *Incorporate the cost of switching connections.* One focus of our algorithm presented in this work is utilizing the diversity of the network by switching possible source peer. In an unstructured P2P network, a search phase is often required before a new connection can be made. Searching for new source peers requires time and incur overhead traffic. In file-sharing applications, the time required to establish new connections is relatively small compare to the typical long session time [96]. Further, due to the fact that the lifetime of peers in P2P file-sharing applications is typically very long, a downloading peer can often actively switch its source peers. Peers switch its source peers actively are able to do a graceful handoff, i.e. negotiate a new connection before disconnecting

a current link, hence reducing the impact of switching peers (the flow of data is not likely to be interrupted). On the other hand, the typical session time in streaming applications is often very short and over 70% of the sessions are less than 1 minute [97]. Hence, peers may be forced to switch their source peers due to their source peers departing the network. This characteristics of short-lived streaming sessions introduce new challenges because the time period that is required for establish new connections, during which the downloading peer receives nothing from the network, have an impact on the performance. Although many modern streaming softwares implements playback buffers to combat connection disruptions. Having playback buffers alone does not solve the issue of playback stutters in real world. To recover from a connection loss, a peer selection strategy has to be able to find a source peer that offering data rate larger than the required minimum streaming rate. It is obvious that the longer we wait for the search algorithm to return possible source peers, the more likely that we can find a peer offering large data rate. However, the longer we wait, the more likely we are going to experience a buffer underflow and hence an interruption in playback. Hence, a good peer selection strategy for streaming applications should consider the “cost” for switching peers.

- 2 *Combine lifetime and capacity information.* Recall that streaming applications are sensitive to connection loss. It is well known that a peer tends to stay in the system longer if it has already been in the system for a while [79, 27]. In other words, the “age” information of a peer can be used to predict whether it will depart the network soon. Several studies try to minimize the probability of connection loss (or node isolation) of a downloading peer due to its neighboring peers departing the network by utilizing the age information of its neighboring peers. However, connection loss is often inevitable when most of its neighboring peers have short lifetime. Hence, another approach to minimize the impact of a connection loss is to maximize the rate at which playback buffer is refilled after a peer experiences a connection loss. As we mentioned in the previous paragraph, searching and establishing connections to new peers require time. The time that a source peer can contribute its capacity is its residual lifetime in the network. Considering the fact that searching for new peers takes time, the “effective” capacity that a downloading peer receives from a source

peer is prorated by the time period spent on searching. Therefore, a downloading peer should consider the capacity as well as the residual lifetime of a possible peer when making a selection decision. For example, if we have to choose one peer from the two peers offering the same upload capacity of 10Mbps. One of them has a residual lifetime of 1 minute while the other has a residual lifetime of 30 minutes. The obvious choice is the one with longer lifetime. From the simple example above, it is important that we incorporate both the lifetime information and the capacity information when designing a peer selection strategy for streaming applications.

Bibliography

- [1] Bram Cohen. *BitTorrent Protocol Specification*.
- [2] The Gnutella Developer Forum. *The Annotated Gnutella Protocol Specification v0.4*.
- [3] Yoram Kulbak, Scott Kirkpatrick, and Danny Bickson. The emule protocol specification. Technical report, The Hebrew University of Jerusalem, 2005.
- [4] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In *Proceedings of IEEE Infocom*, 2005.
- [5] Tribler. <http://www.tribler.org/>.
- [6] Sopcast. <http://www.sopcast.org/>.
- [7] Skype. <http://www.skype.com/>.
- [8] Seti@home. <http://setiathome.berkeley.edu/>.
- [9] Yacy. <http://sciencenet.fzk.de/>.
- [10] T. Karagiannis, A. Broido, and N. Brownlee. Is p2p dying or just hiding? In *Proceedings of IEEE Globecom*, Nov. 2004.
- [11] Alok Madhukar and Carey Williamson. A longitudinal study of p2p traffic classification. In *IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS)*, pages 179–188, Washington, DC, USA, 2006. IEEE Computer Society.

- [12] Jeffrey Eрман, Anirban Mahanti, Martin Arlitt, and Carey Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *International conference on World Wide Web*, pages 883–892, New York, NY, USA, 2007. ACM.
- [13] Hui Liu, Wenfeng Feng, Yongfeng Huang, and Xing Li. A peer-to-peer traffic identification method using machine learning. In *IEEE International Conference on Networking, Architecture, and Storage*, 2007.
- [14] D. Qiu and R. Srikant. Modelling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of ACM Sigcomm*, Aug. 2004.
- [15] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *Proceedings of IEEE Infocom*, March 2004.
- [16] X. Yang and G. de Veciana. Performance of peer-to-peer networks: Service capacity and role of resource sharing policies. *Performance Evaluation - Special Issue on Performance Modeling and Evaluation of P2P Computing Systems*, 63(3):175–194, March 2006.
- [17] S. Saroiu, K. P. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of ACM Multimedia Computing and Networking (MMCN)*, 2002.
- [18] K. P. Gummadi, R. J. Dunn, and S. Saroiu. Measurement, modeling, and analysis of a peer-to-peer file sharing workload. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2003.
- [19] D. S. Bernstein, Z. Feng, and B. N. Levine. Adaptive peer selection. In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, February 2003.
- [20] M. Adler, R. Kumar, K. Ross, D. Rubenstein, D. Turner, and D. D. Yao. Optimal peer selection in a free-market peer-resource economy. In *Workshop on Economics of Peer-to-Peer Systems*, Cambridge, MA, Jun. 2004.

- [21] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, and D.D. Yao. Optimal peer selection for p2p downloading and streaming. In *Proceedings of IEEE Infocom*, volume 3, pages 1538–1549 vol. 3, March 2005.
- [22] S. G. M. Koo, K. Kannan, and C. S. G. Lee. Neighbor-selection strategy in peer-to-peer networks. In *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN)*, Rosemont, IL, Oct. 2004.
- [23] Y. M. Chiu and D. Y. Eun. Minimizing file download time over stochastic channels in peer-to-peer networks. In *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2006.
- [24] Y. M. Chiu and D. Y. Eun. On the performance of download strategies in a p2p like network. In *IEEE Globecom*, Washington, DC, Nov 2007.
- [25] Y. M. Chiu and D. Y. Eun. Minimizing file download time in stochastic peer-to-peer networks. *IEEE/ACM Transactions on Networking*, 16(2):253–266, April 2008.
- [26] Z. Yao and D. Loguinov. Link lifetimes and randomized neighbor selection in dhds. In *Proceedings of IEEE Infocom*, pages 146–150, April 2008.
- [27] Feng Wang, Jiangchuan Liu, and Yongqiang Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of IEEE Infocom*, Phoenix, AZ, April 2008.
- [28] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of ACM Conference on Electronic Commerce*, 2001.
- [29] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.
- [30] S. Jun and M. Ahamad. Incentives in BitTorrent induce free riding. In *ACM SIGCOMM workshop on Economics of peer-to-peer systems*, Philadelphia, Pennsylvania, August 2005.
- [31] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In *Workshop on Hot Topics in Networks (HotNets)*, Irvine, CA, November 2006.

- [32] Kolja Eger and Ulrich Killat. Fair resource allocation in peer-to-peer networks. *Computer Communications*, 30(16):3046–3054, June 2007.
- [33] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in BitTorrent systems. In *Proceedings of ACM Sigmetrics*, San Diego, CA, USA, June 2007.
- [34] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in BitTorrent? In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Cambridge, MA, April 2007.
- [35] T. Locher, S. Schmid, and R. Wattenhofer. Rescuing tit-for-tat with source coding. In *IEEE International Conference on Peer-to-Peer Computing (P2P'07)*, pages 3–10, Washington, DC, USA, 2007. IEEE Computer Society.
- [36] Kolja Eger and Ulrich Killat. Bandwidth trading in BitTorrent-like p2p networks for content distribution. *Computer Communications*, 31(2):201–211, 2008.
- [37] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. BitTorrent is an auction: Analyzing and improving BitTorrent’s incentives. In *Proceedings of ACM Sigcomm*, SEATTLE, WA, August 2008. ACM.
- [38] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, UC Berkeley, Berkeley, CA, 2000.
- [39] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platform*, pages 329–350, Heidelberg, Germany, November 2001.
- [40] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM Sigcomm*, 2001.
- [41] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *Proceedings of ACM Sigcomm*, 2002.

- [42] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.
- [43] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 5, Washington, DC, USA, 2002. IEEE Computer Society.
- [44] Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, and Frans F. Kaashoek. Comparing the performance of distributed hash tables under churn. In *Proc. of the 3rd International Workshop on Peer-to-Peer Systems*, February 2004.
- [45] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.
- [46] R. H. Wouhaybi and A. T. Campbell. Phenix: supporting resilient low-diameter peer-to-peer topologies. In *Proceedings of IEEE Infocom*, volume 1, page 119, 2004.
- [47] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom*, Miami, FL, March 2005.
- [48] L. Massoulie and M. Vojnovic. Coupon replication systems. In *In Proc. of ACM SIGMETRICS*, pages 2–13. ACM Press, 2005.
- [49] R. Kumar, Yong Liu, and K. Ross. Stochastic fluid theory for p2p streaming systems. In *Proceedings of IEEE Infocom*, pages 919–927, May 2007.
- [50] M. Lin, B. Fan, J. C. S. Lui, and D. M. Chiu. Stochastic analysis of file-swarming systems. *Perform. Eval.*, 64(9-12):856–875, 2007.
- [51] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *Proceedings of ACM Sigmetrics*, pages 325–336, New York, NY, USA, 2008. ACM.

- [52] S. Koo, C. Rosenberg, and D. Xu. Analysis of parallel downloading for large file distribution. In *Proceedings of IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS)*, May 2003.
- [53] F. Lo Piccolo, G. Neglia, and G. Bianchi. The effect of heterogeneous link capacities in BitTorrent-like file sharing system. In *IEEE International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, Oct. 2004.
- [54] K. K. Ramachandran and B. Sikdar. An analytic framework for modeling peer to peer networks. In *Proceedings of IEEE Infocom*, March 2005.
- [55] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang. Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In *Proceedings of IEEE Infocom*, April 2003.
- [56] C. Gkantsidis, M. Ammar, and E. Zegura. On the effect of large-scale deployment of parallel downloading. In *Proceedings of IEEE Workshop on Internet Applications (WIAPP)*, Jun. 2003.
- [57] Y. Kulbak and D. Bickson. *The eMule Protocol Specification*, Jan. 2005. “<http://leibniz.cs.huji.ac.il/tr/acc/2005/HUJI-CSE-LTR-2005-3/emule.pdf>”.
- [58] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *infocom*, Hong Kong, March 2004.
- [59] The network simulator ns-2.
- [60] Brad Stone. As web traffic grows, crashes take bigger toll, June 2008. http://www.nytimes.com/2008/07/06/technology/06outage.html?_r=1.
- [61] Clay Shirky. What is p2p... and what isn't?, November 2000. <http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html>.
- [62] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computer Surveys*, 36(4):335–371, December 2004.
- [63] Ellacoya Sharman Networks. *Kazaa*. Kazaa.

- [64] Mojonation: an internet distributed content system with market based resource allocation. <http://sourceforge.net/projects/mojonation/>.
- [65] John Kubiawicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.
- [66] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 67–95, 2000.
- [67] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, Second Quarter 2005.
- [68] Nima Sarshar, P. Oscar Boykin, and Vwani P. Roychowdhury. Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing*, pages 2–9, Washington, DC, USA, 2004. IEEE Computer Society.
- [69] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file sharing communities. In *Proceedings of IEEE Infocom*, 2004.
- [70] Hasan Guclu and Murat Yuksel. Scale-free overlay topologies with hard cutoffs for unstructured peer-to-peer networks. In *International Conference on Distributed Computing Systems (ICDCS 07)*, page 32, Washington, DC, USA, 2007. IEEE Computer Society.
- [71] H. Guclu, D. Kumari, and M. Yuksel. Ad hoc limited scale-free models for unstructured peer-to-peer networks. In *P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, pages 160–169, Washington, DC, USA, 2008. IEEE Computer Society.

- [72] Kin-Wah Kwong and Danny H. K. Tsang. Building heterogeneous peer-to-peer networks: Protocol and analysis. *IEEE/ACM Transactions on Networking*, 16(2):281–292, April 2008.
- [73] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of ACM Sigcomm*, pages 161–172, New York, NY, USA, 2001. ACM.
- [74] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, 1997.
- [75] Albert-Laszlo Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.
- [76] D. Leonard, V. Rai, and D. Loguinov. On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. In *Proceedings of ACM Sigmetrics*, June 2005.
- [77] Derek Leonard, Zhongmei Yao, Xiaoming Wang, and Dmitri Loguinov. On static and dynamic partitioning behavior of large-scale networks. In *Proceedings of IEEE International Conference on Network Protocols*, pages 345–357, 2005.
- [78] Z. Yao, D. Leonard, X. Wang, and D. Loguinov. Modeling heterogeneous user churn and local resilience of unstructured p2p networks. In *Proceedings of IEEE International Conference on Network Protocols*, November 2006.
- [79] Z. Yao, X. Wang, D. Leonard, and D. Loguinov. On node isolation under churn in unstructured p2p networks with heavy-tailed lifetimes. In *Proceedings of IEEE Infocom*, May 2007.
- [80] S. Buchegger, J. Munding, and J. Y. Le Boudec. Reputation systems for self-organized networks. *IEEE Technol. Soc. Mag.*, 27(1):41–47, Spring 2008.
- [81] R. Bhattacharjee and A. Goel. Avoiding ballot stuffing in ebay-like reputation systems. In *Workshop on Economics of Peer-to-Peer Systems*. ACM, Aug 2005.

- [82] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Workshop on Economics of Peer-to-Peer Systems*. ACM, Aug 2005.
- [83] A. Ghose, P. Ipeirotis, and A. Sundararajan. Reputation premium and network structure in electronic peer-to-peer markets: Analyzing textual feedback and network structure. In *Workshop on Economics of Peer-to-Peer Systems*. ACM, Aug 2005.
- [84] R. Jurca and B. Faltings. Reputation-based pricing of p2p services. In *Workshop on Economics of Peer-to-Peer Systems*. ACM, Aug 2005.
- [85] J. Mundinger and J. Y. Le Boudec. Analysis of a reputation system for mobile ad-hoc networks with liars. In *Proc. Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks WIOPT 2005*, pages 41–46, 3–7 April 2005.
- [86] K. Walsh and E. G. Sirer. Combating peer-to-peer spam and decoys with object reputation. In *Workshop on Economics of Peer-to-Peer Systems*. ACM, Aug 2005.
- [87] D. Purandare and R. Guha. Preferential and strata based p2p model: selfishness to altruism and fairness. In *IEEE International Conference on Parallel and Distributed Systems*, volume 1, pages 8 pp.–, 0-0 2006.
- [88] Kolja Eger and Ulrich Killat. Resource pricing in Peer-to-Peer networks. *IEEE Communications Letters*, 11(1):82–84, 2007.
- [89] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, Aug. 2003.
- [90] M. Jain and C. Dovrolis. End-to-end estimation of the available bandwidth variation range. In *Proceedings of ACM Sigmetrics*, Jun. 2005.
- [91] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *Proceedings of ACM Sigcomm*, Pittsburgh, PA, 2002.
- [92] S. M. Ross. *Stochastic Processes*. John Wiley & Son, New York, second edition, 1996.

- [93] I. Rhee and L. Xu. Limitations of equation-based congestion control. In *Proceedings of ACM Sigcomm*, Aug. 2005.
- [94] A. Müller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks*. John Wiley & Son, New York, NY, 2002.
- [95] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, USA, 1995.
- [96] Intel. Peer-to-peer content distribution: Using client pc resources to store and distribute content in the enterprise. Technical report, Intel, September 2003.
- [97] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching television over an IP network. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 71–84, New York, NY, USA, 2008. ACM.