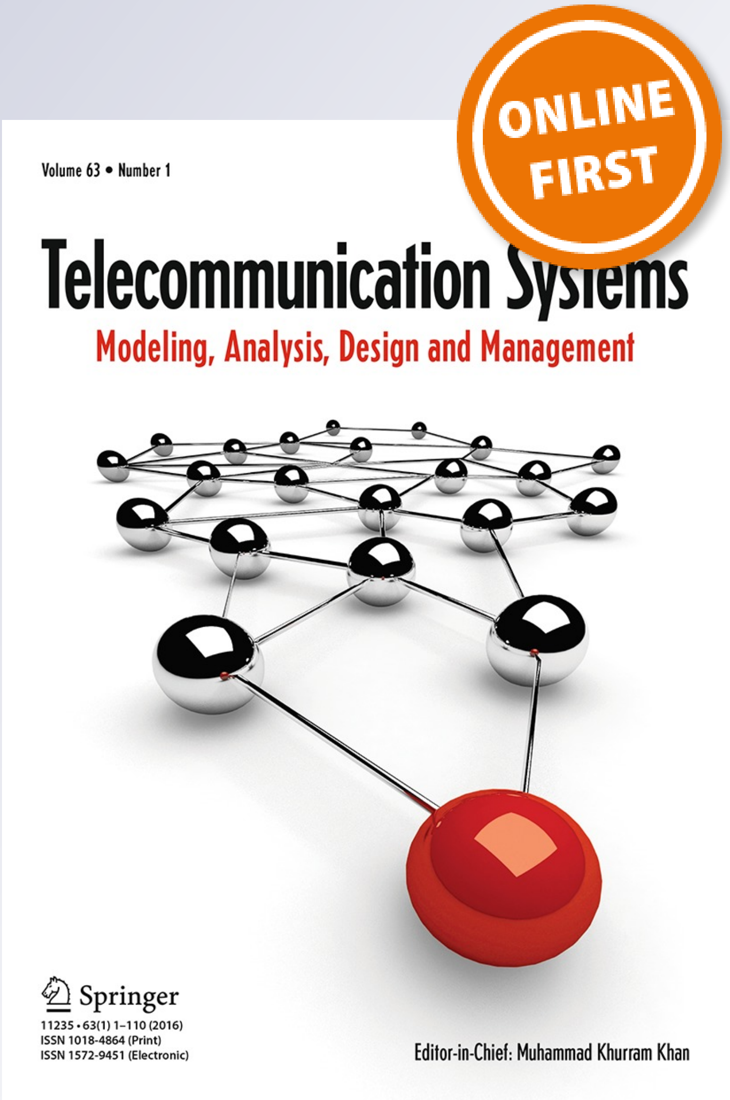
Design and analysis of parallel file downloading algorithms in peer-to-peer networks

Keqin Li

Telecommunication Systems
Modelling, Analysis, Design and Management

ISSN 1018-4864

Telecommun Syst
DOI 10.1007/s11235-016-0203-1



Telecommun Syst
DOI 10.1007/s11235-016-0203-1

Design and analysis of parallel file downloading algorithms in peer-to-peer networks

Keqin Li[1]

Abstract It is well known that the method of parallel downloading can be used to reduce file download times in a peer-to-peer (P2P) network. There has been little investigation on parallel download and chunk allocation for source peers with random service capacities. The main contribution of this paper is to address the problem of efficient parallel file download in P2P networks with random service capacities. A precise analysis of the expected download time is given when the service capacity of a source peer is a random variable. A general framework is developed for analyzing the expected download time of a parallel download and chunk allocation algorithm, and is applied to the analysis of several algorithms. Two chunk allocation algorithms for parallel download are proposed. It is observed that the performance of parallel download can be significantly improved by using the method of probing high-capacity peers. One such algorithm is proposed and its expected parallel download time is analyzed. The performance of these parallel file download algorithms in P2P networks with random service capacities are compared. The above parallel download algorithms are extended to multiple file download by dividing source peers into clusters. It is noticed that there is an important issue of optimal parallelism which minimizes the combined effect of intracluster and intercluster overhead of parallel download and load imbalance.

Keywords Chunk allocation · Download time · File sharing system · Parallel downloading · Peer-to-peer network · Random service capacity

✉ Keqin Li
lik@newpaltz.edu

[1] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

1 Introduction

A peer-to-peer (P2P) network is able to provide various services by employing diverse connectivity among participating peers and the combined resources of participants including storage space, computing power, and communication bandwidth [3]. There are two significant advantages in P2P networks. The first advantage is scalability, i.e., the total service capacity of a P2P network increases as participating peers in the network increases. An important feature of a P2P network is that all peers contribute resources, and all peers function as both clients and servers. The second advantage is reliability, i.e., the robustness and fault tolerance of a P2P network increases due to the distributed nature of the network and the capability of replicating data over multiple peers. In a pure P2P network, peers find locations of data without relying on a centralized index server, which means that there is no single point of failure in the network. File sharing using application layer protocols such as BitTorrent is the most popular application of P2P networks. Current popular file sharing networks and services include Gnutella/Gnutella2 (G2), eDonkey, Direct Connect, BitTorrent, RetroShare, Mininova, isoHunt, The Pirate Bay, and KickassTorrents [2].

Extensive investigation has been performed by many researchers in the last few years for performance measurement, modeling, analysis, and optimization of file sharing in P2P networks. Research in this area has been conducted at three different levels, i.e., system level, peer group level, and individual peer level. At the system level, research is focused on establishing models of P2P networks such as queueing models [13,30] and fluid models [12], so that overall system characterizations such as system throughput and average file download time can be obtained. At the peer group level, research is focused on distributing a file from a set of source

Published online: 26 July 2016

peers to a set of user peers, so that the overall distribution time is minimized [16,18,23,24,29,31]. At the individual peer level, research is focused on analyzing and minimizing the file download time of a single peer [10,19,21,22]. It is clear that the vast majority of file downloads are performed by individual users. Therefore, P2P network performance optimization from a single peer's point of view has been an interesting and important issue.

File download strategies for an individual user peer can be classified into two categories, namely, single download methods from one source peer and parallel download methods from several source peers simultaneously. The main concern in single download methods is the peer selection problem, namely, switching among source peers and finally settling on one, while keeping the total time of probing and downloading to a minimum [4–6,8,11,19,21,22]. It is well known that the method of parallel downloading can be used to reduce file download times. The main concern in parallel download methods is the chunk allocation problem, namely, how to divide a file to be downloaded into chunks which can be downloaded from several source peers simultaneously. In [10], it is proposed that a file is divided into chunks of equal sizes. In [27], it is observed that to achieve the maximum speedup, chunks should be allocated such that all servers finish their transmissions at the same time. It has been observed that performance improvement experienced by clients who perform parallel downloading comes at the expense of clients who simply go to a single server to retrieve files [14].

Performance measurement, modeling, analysis, and optimization of parallel document downloading in the Internet and file sharing in P2P networks have also been conducted at three different levels, i.e., system level, peer/client group level, and individual peer/client level. At the system level, research is focused on understanding the impact of large scale parallel downloading on the performance of a network [14,17,25,26]. At the peer/client group level, research is focused on parallel document/file downloading from multiple mirror sites and source peers such that duplicated transmissions are kept to a minimum by using efficient multicasting [7]. At the individual peer/client level, research is focused on minimizing the parallel file download time for a single peer/client [10,27]. Fine parallel downloading algorithms for an individual user peer is critical in competing for network resources. However, there is lack of comprehensive and analytical performance study of parallel download algorithms, especially when source peers have random service capacities [20].

The main contribution of this paper is to address the problem of efficient parallel file download in peer-to-peer networks with random service capacities. We give a precise analysis of the expected download time when the service capacity of a source peer is a random variable (Sect. 2). We develop a general framework for analyzing the expected download time of a parallel download and chunk allocation algorithm (Sect. 3.1), and apply the framework to the analysis of several algorithms (Sects. 3.2–3.4). We propose two chunk allocation algorithms for parallel download (Sects. 3.3–3.4). We observe that the performance of parallel download can be significantly improved by using the method of probing high-capacity peers. We propose such an algorithm and analyze the expected download time of our algorithm (Sect. 3.5). We compare the performance of these parallel file download algorithms in P2P networks with random service capacities (Sect. 4). We also extend the above parallel download algorithms to multiple file download by dividing source peers into clusters and analyze the expected download time (Sect. 5.1). We notice that there is an important issue of optimal parallelism which minimizes the combined effect of intracluster and intercluster overhead of parallel download and load imbalance (Sect. 5.2).

2 Preliminaries

Throughout the paper, we use $\boldsymbol{P}[e]$ to denote the probability of an event e, $f_X(x)$ the probability distribution function (pdf), $F_X(x)$ the cumulative distribution function (cdf), and $\boldsymbol{E}(X)$ the expectation, respectively, of a random variable X.

Assume that n peers $1, 2, \ldots, n$ have been identified as source peers of a file of interest, such that any part of the file can be downloaded from any of these n source peers. We further assume that the service capacity (i.e., the download speed experienced by a user, or the number of bits that can be downloaded in one unit of time measured in, e.g., kbps, MB/min) of source peer i is C_i, a random variable in $[0, \infty)$ with pdf $f_{C_i}(c)$ and cdf $F_{C_i}(c)$. (Notice that if C_i is in a finite range $[0, B]$, we simply have $f_{C_i}(c) = 0$ for all $c > B$.) It is also assumed that all source peers are stable, i.e., they remain in a P2P network for significant amount of time. In other words, there is no effect of peer churn [28] for downloading the file of interest, i.e., all source peers are available during downloading of the file. Moreover, all the n source peers are seed peers, i.e., they all hold a complete copy of a file, such that any chunk of the file can be obtained from any source peer. Further analysis of the effect of peer churn and non-seed peers can be a direction for future investigation.

We use S to represent the size as well as the name of a file. The file size is measured in the number of units of data, where one unit of data can be kilo-byte (KB) or mega-byte (MB). Let $T_i(S)$ be the download time of a file of size S from source peer i. The following theorem gives the expected download time of a complete file (or any chunk of a file) of size S from source peer i.

Theorem 1 *The expected download time of a file of size S from source peer i is*

$$\boldsymbol{E}(T_i(S)) = S\boldsymbol{E}(T_i(1)),$$

where

$$\boldsymbol{E}(T_i(1)) = \int_0^\infty \frac{f_{C_i}(c)}{c} dc$$

is the expected download time of one unit of data from source peer i.

Proof It is clear that

$$T_i(S) = \frac{S}{C_i}.$$

Let $T_i(S, c) = S/c$ be the download time of a file of size S from source peer i when $C_i = c$. The first equation in the theorem can be obtained by randomizing c in $T_i(S, c)$, i.e.,

$$\boldsymbol{E}(T_i(S)) = \int_0^\infty T_i(S, c) f_{C_i}(c) dc = \int_0^\infty \frac{S}{c} f_{C_i}(c) dc.$$

The above equation can also be written as

$$\boldsymbol{E}(T_i(S)) = S \int_0^\infty \frac{f_{C_i}(c)}{c} dc = S\boldsymbol{E}(T_i(1)),$$

that is, $\boldsymbol{E}(T_i(S))$ is a linear function of S, where

$$\boldsymbol{E}(T_i(1)) = \int_0^\infty \frac{f_{C_i}(c)}{c} dc$$

is the expected download time of one unit of data from source peer i. □

We say that the n source peers are homogeneous if their service capacities $C_1, C_2, \ldots, C_n$ are identical random variables C with the same pdf, i.e.,

$$f_{C_1}(c) = f_{C_2}(c) = \cdots = f_{C_n}(c) = f_C(c),$$

and the same cdf, i.e.,

$$F_{C_1}(c) = F_{C_2}(c) = \cdots = F_{C_n}(c) = F_C(c).$$

Notice that this does not mean that the n source peers have the same service capacity. In fact, during transferring the same file at the same time, the service capacities of the n source peers can be entirely and radically different as governed by $f_C(c)$.

For homogeneous source peers, we use $\boldsymbol{E}(T(S)) = S\boldsymbol{E}(T(1))$ to represent the expected download time of a file of size S from any source peer, where

$$\boldsymbol{E}(T(1)) = \int_0^\infty \frac{f_C(c)}{c} dc$$

is the expected download time of one unit of data from any source peer.

3 Parallel download and chunk allocation

A file can be downloaded from r service peers $1, 2, \ldots, r$ simultaneously by using a parallel download algorithm. For instance, in BitTorrent, a file is divided into small chunks, and a peer can download multiple chunks of the file from different source peers simultaneously [9,15]. It is assumed that the r source peers are independent and do not interfere with each other. Although in reality, this might not be the case, especially when a number of servers share network paths and thus have correlated service capacities.

3.1 A generic algorithm and analysis

3.1.1 Algorithm

In a generic parallel download algorithm A, a file of size S is divided into r chunks of sizes $S_1, S_2, \ldots, S_r$, such that chunk S_i is downloaded from source peer i, for all $1 \le i \le r$ in parallel. Different algorithms have different strategies in choosing the chunk sizes and have different parallel download times. The key issue is then to choose the best chunk sizes.

Notice that in a real communication, when a packet is sent, additional information is needed (e.g., the header of a packet). Therefore, splitting a file into chunks results in a slightly different total size of the original file due to such additional information in each chunk. However, we believe that the size of such additional data is negligible compared to the size of a chunk, and the impact of such overhead is not included in our discussion.

3.1.2 Analysis

Let $T_A(S, r)$ denote the parallel download time of algorithm A for a file of size S from r source peers with chunk sizes $S_1, S_2, \ldots, S_r$. The following theorem gives the expected parallel download time of algorithm A.

Theorem 2 *The expected parallel download time of algorithm A is*

$$\boldsymbol{E}(T_A(S, r)) = \int_0^\infty \left(1 - \prod_{i=1}^r \left(1 - F_{C_i}\left(\frac{S_i}{t}\right)\right)\right) dt,$$

or, equivalently,

$$\boldsymbol{E}(T_A(S, r)) = \sum_{i=1}^r \int_0^\infty \frac{S_i}{c} f_{C_i}(c) \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{S_i} c\right)\right) dc.$$

Proof The parallel download time of algorithm A with chunk sizes $S_1, S_2, \ldots, S_r$ for a file of size S from r source peers is

$$T_A(S,r) = \max\{T_1(S_1), T_2(S_2), \ldots, T_r(S_r)\}.$$

Since $T_i(S_i) = S_i/C_i$, we get the cdf of $T_i(S_i)$ as follows,

$$\begin{aligned} F_{T_i(S_i)}(t) &= \boldsymbol{P}[T_i(S_i) \le t] \\ &= \boldsymbol{P}\left[\frac{S_i}{C_i} \le t\right] \\ &= \boldsymbol{P}\left[C_i \ge \frac{S_i}{t}\right] \\ &= 1 - F_{C_i}\left(\frac{S_i}{t}\right), \end{aligned}$$

and the pdf of $T_i(S_i)$ as follows,

$$f_{T_i(S_i)}(t) = \frac{S_i}{t^2} f_{C_i}\left(\frac{S_i}{t}\right),$$

for all $t > 0$. Hence, the cdf of $T_A(S,r)$ is

$$\begin{aligned} F_{T_A(S,r)}(t) &= \boldsymbol{P}[T_A(S,r) \le t] \\ &= \prod_{i=1}^{r} \boldsymbol{P}\left[T_i(S_i) \le t\right] \\ &= \prod_{i=1}^{r} F_{T_i(S_i)}(t) \\ &= \prod_{i=1}^{r} \left(1 - F_{C_i}\left(\frac{S_i}{t}\right)\right), \end{aligned}$$

and the pdf of $T_A(S,r)$ is

$$f_{T_A(S,r)}(t) = \sum_{i=1}^{r} \frac{S_i}{t^2} f_{C_i}\left(\frac{S_i}{t}\right) \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{t}\right)\right),$$

for all $t > 0$. The expected parallel download time of algorithm A is

$$\begin{aligned} \boldsymbol{E}(T_A(S,r)) &= \int_0^\infty (1 - F_{T_A(S,r)}(t))dt \\ &= \int_0^\infty \left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}\left(\frac{S_i}{t}\right)\right)\right)dt, \end{aligned}$$

or, equivalently,

$$\begin{aligned} &\boldsymbol{E}(T_A(S,r)) \\ &\quad = \int_0^\infty t f_{T_A(S,r)}(t)dt \\ &\quad = \int_0^\infty t \sum_{i=1}^{r} \frac{S_i}{t^2} f_{C_i}\left(\frac{S_i}{t}\right) \prod_{i' \ne i}\left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{t}\right)\right) dt \\ &\quad = \sum_{i=1}^{r} \int_0^\infty \frac{S_i}{t} f_{C_i}\left(\frac{S_i}{t}\right) \prod_{i' \ne i}\left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{t}\right)\right) dt \\ &\quad = \sum_{i=1}^{r} \int_\infty^0 c f_{C_i}(c) \prod_{i' \ne i}\left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{S_i}c\right)\right) \\ &\qquad \times \left(-\frac{S_i}{c^2}\right) dc \quad \left(\text{by letting } c = \frac{S_i}{t}\right) \\ &\quad = \sum_{i=1}^{r} \int_0^\infty \frac{S_i}{c} f_{C_i}(c) \prod_{i' \ne i}\left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{S_i}c\right)\right) dc. \end{aligned}$$

This proves the theorem. □

Our main problem here is to find chunk sizes $S_1, S_2, \ldots, S_r$, such that the expected parallel download time $\boldsymbol{E}(T_A(S,r))$ is minimized. This is a well defined multi-variable optimization problem. Unfortunately, the problem is very complicated to solve, even numerically. In this paper, we will analyze several simple heuristic solutions to the problem.

3.2 Algorithm $\mathbb{PD}_0$ and analysis

3.2.1 Algorithm

In the naive parallel download algorithm $\mathbb{PD}_0$, a file of size S is divided into r chunks of equal size, i.e., $S_1 = S_2 = \cdots = S_r = S/r$ [10]. Algorithm $\mathbb{PD}_0$ has no knowledge of and does not probe the current service capacities of the source peers.

3.2.2 Analysis

The following theorem gives the expected parallel download time of algorithm $\mathbb{PD}_0$ for a file of size S from r source peers.

Theorem 3 *The expected parallel download time of algorithm $\mathbb{PD}_0$ is*

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(S,r)) = S\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)),$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \frac{1}{r}\int_0^\infty \frac{1}{c^2}\left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}(c)\right)\right)dc,$$

or, equivalently,

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \frac{1}{r}\sum_{i=1}^{r}\int_0^\infty \frac{f_{C_i}(c)}{c}\prod_{i' \ne i}\left(1 - F_{C_{i'}}(c)\right)dc,$$

is the expected download time of one unit of data.

Proof The parallel download time of algorithm $\mathbb{PD}_0$ for a file of size S from r source peers is

$$T_{\mathbb{PD}_0}(S,r) = \max\left\{T_1\left(\frac{S}{r}\right), T_2\left(\frac{S}{r}\right), \ldots, T_r\left(\frac{S}{r}\right)\right\}.$$

The cdf of $T_{\mathbb{PD}_0}(S,r)$ is

$$F_{T_{\mathbb{PD}_0}(S,r)}(t) = \prod_{i=1}^{r}\left(1 - F_{C_i}\left(\frac{S}{rt}\right)\right),$$

and the pdf of $T_{\mathbb{PD}_0}(S,r)$ is

$$f_{T_{\mathbb{PD}_0}(S,r)}(t) = \sum_{i=1}^{r}\frac{S}{rt^2}f_{C_i}\left(\frac{S}{rt}\right)\prod_{i'\neq i}\left(1 - F_{C_{i'}}\left(\frac{S}{rt}\right)\right),$$

for all $t > 0$.

The expected parallel download time of algorithm $\mathbb{PD}_0$ is

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_0}(S,r)) &= \int_0^\infty (1 - F_{T_{\mathbb{PD}_0}(S,r)}(t))dt \\
&= \int_0^\infty \left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}\left(\frac{S}{rt}\right)\right)\right)dt \\
&= \int_\infty^0 \left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}(c)\right)\right) \\
&\quad \times \left(-\frac{S}{rc^2}\right)dc \quad \left(\text{by letting } c = \frac{S}{rt}\right) \\
&= \int_0^\infty \frac{S}{rc^2}\left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}(c)\right)\right)dc \\
&= S\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \frac{1}{r}\int_0^\infty \frac{1}{c^2}\left(1 - \prod_{i=1}^{r}\left(1 - F_{C_i}(c)\right)\right)dc,$$

or, equivalently, by Theorem 2,

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_0}(S,r)) &= \sum_{i=1}^{r}\int_0^\infty \frac{S}{rc}f_{C_i}(c)\prod_{i'\neq i}\left(1 - F_{C_{i'}}(c)\right)dc \\
&= S\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \frac{1}{r}\sum_{i=1}^{r}\int_0^\infty \frac{f_{C_i}(c)}{c}\prod_{i'\neq i}\left(1 - F_{C_{i'}}(c)\right)dc.$$

This proves the theorem. □

For homogeneous source peers, we get the cdf of $T_{\mathbb{PD}_0}(S,r)$,

$$F_{T_{\mathbb{PD}_0}(S,r)}(t) = \left(1 - F_C\left(\frac{S}{rt}\right)\right)^r,$$

and the pdf of $T_{\mathbb{PD}_0}(S,r)$,

$$f_{T_{\mathbb{PD}_0}(S,r)}(t) = \frac{S}{t^2}f_C\left(\frac{S}{rt}\right)\left(1 - F_C\left(\frac{S}{rt}\right)\right)^{r-1},$$

for all $t > 0$. The expected parallel download time of algorithm $\mathbb{PD}_0$ is

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_0}(S,r)) &= \int_0^\infty \frac{S}{rc^2}\left(1 - (1 - F_C(c))^r\right)dc \\
&= S\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \frac{1}{r}\int_0^\infty \frac{1}{c^2}\left(1 - (1 - F_C(c))^r\right)dc,$$

or, equivalently,

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_0}(S,r)) &= \int_0^\infty \frac{S}{c}f_C(c)\,(1 - F_C(c))^{r-1}dc \\
&= S\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) = \int_0^\infty \frac{f_C(c)}{c}\,(1 - F_C(c))^{r-1}dc.$$

3.3 Algorithm $\mathbb{PD}_1$ and analysis

3.3.1 Algorithm

Our algorithm $\mathbb{PD}_1$ for parallel download and chunk allocation without probing works as follows. Instead of dividing a file into chunks of equal sizes, algorithm $\mathbb{PD}_1$ divides a file of size S into chunks of sizes $S_1, S_2, \ldots, S_r$, such that all the r source peers have the same expected download time, i.e.,

$$\boldsymbol{E}(T_1(S_1)) = \boldsymbol{E}(T_2(S_2)) = \cdots = \boldsymbol{E}(T_r(S_r)).$$

Algorithm $\mathbb{PD}_1$ has no knowledge of and does not probe the current service capacities of the source peers. However, algorithm $\mathbb{PD}_1$ attempts to do chunk allocation based on the expected behavior of source peers (e.g., the expected download time of one unit of data, which is certainly available, since we assume that the pdf of each source peer is known).

Since $\boldsymbol{E}(T_i(S_i)) = S_i\boldsymbol{E}(T_i(1))$, for all $1 \le i \le r$, where $\boldsymbol{E}(T_i(1))$ is given by Theorem 1, we have

$$S_1\boldsymbol{E}(T_1(1)) = S_2\boldsymbol{E}(T_2(1)) = \cdots = S_r\boldsymbol{E}(T_r(1)) = T,$$

for some T, which implies that

$$S_i = \frac{T}{\boldsymbol{E}(T_i(1))}.$$

Since $S_1 + S_2 + \cdots + S_r = S$, we have

$$\frac{T}{\boldsymbol{E}(T_1(1))} + \frac{T}{\boldsymbol{E}(T_2(1))} + \cdots + \frac{T}{\boldsymbol{E}(T_r(1))} = S,$$

which gives rise to

$$T = S\left(\sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\right)^{-1},$$

and

$$S_i = \frac{S}{\boldsymbol{E}(T_i(1))}\left(\sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\right)^{-1},$$

for all $1 \le i \le r$. In words, each chunk size is proportional to the reciprocal of the expected download time of one unit of data from a source peer.

3.3.2 Analysis

The following theorem gives the expected parallel download time of algorithm $\mathbb{PD}_1$ for a file of size S from r source peers.

Theorem 4 *The expected parallel download time of algorithm $\mathbb{PD}_1$ is*

$$\boldsymbol{E}(T_{\mathbb{PD}_1}(S, r)) = S\boldsymbol{E}(T_{\mathbb{PD}_1}(1, r)),$$

where

$$\begin{aligned}&\boldsymbol{E}(T_{\mathbb{PD}_1}(1, r))\\ &\quad = \left(\sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\right)^{-1} \times \sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\int_0^\infty \frac{f_{C_i}(c)}{c}\\ &\qquad \prod_{i'\ne i}\left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(T_i(1))}{\boldsymbol{E}(T_{i'}(1))}c\right)\right)dc,\end{aligned}$$

is the expected download time of one unit of data.

Proof The parallel download time of algorithm $\mathbb{PD}_1$ for a file of size S from r source peers is

$$T_{\mathbb{PD}_1}(S, r) = \max\{T_1(S_1), T_2(S_2), \ldots, T_r(S_r)\}.$$

By Theorem 2, the expected parallel download time of algorithm $\mathbb{PD}_1$ is

$$\begin{aligned}&\boldsymbol{E}(T_{\mathbb{PD}_1}(S, r))\\ &\quad = \sum_{i=1}^{r}\int_0^\infty \frac{S_i}{c} f_{C_i}(c)\prod_{i'\ne i}\left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{S_i}c\right)\right)dc\\ &\quad = \sum_{i=1}^{r}\int_0^\infty \frac{S_i}{c} f_{C_i}(c)\prod_{i'\ne i}\left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(T_i(1))}{\boldsymbol{E}(T_{i'}(1))}c\right)\right)dc\\ &\quad = \sum_{i=1}^{r}\int_0^\infty \frac{S}{c\boldsymbol{E}(T_i(1))}\left(\sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\right)^{-1} f_{C_i}(c)\\ &\qquad \prod_{i'\ne i}\left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(T_i(1))}{\boldsymbol{E}(T_{i'}(1))}c\right)\right)dc\\ &\quad = S\boldsymbol{E}(T_{\mathbb{PD}_1}(1, r)),\end{aligned}$$

where

$$\begin{aligned}&\boldsymbol{E}(T_{\mathbb{PD}_1}(1, r))\\ &\quad = \left(\sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\right)^{-1} \times \sum_{i=1}^{r}\frac{1}{\boldsymbol{E}(T_i(1))}\int_0^\infty \frac{f_{C_i}(c)}{c}\\ &\qquad \prod_{i'\ne i}\left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(T_i(1))}{\boldsymbol{E}(T_{i'}(1))}c\right)\right)dc.\end{aligned}$$

This proves the theorem. □

For r source peers with $\boldsymbol{E}(T_1(1)) = \boldsymbol{E}(T_2(1)) = \cdots = \boldsymbol{E}(T_r(1))$ (including the case of homogeneous source peers), algorithm $\mathbb{PD}_1$ works in exactly the same way as algorithm $\mathbb{PD}_0$, namely, dividing S into r chunks of equal size S/r.

3.4 Algorithm $\mathbb{PD}_2$ and analysis

3.4.1 Algorithm

Algorithm $\mathbb{PD}_2$ for parallel download and chunk allocation without probing works as follows. Instead of allocating chunks to source peers in proportion to the reciprocals of their expected download times of one unit of data, chunk sizes are proportional to the expected service capacities, that is,

$$S_i = \left(\frac{\boldsymbol{E}(C_i)}{\boldsymbol{E}(C_1) + \boldsymbol{E}(C_2) + \cdots + \boldsymbol{E}(C_r)}\right)S,$$

for all $1 \le i \le r$. Since the expected download time of one unit of data and the expected service capacity of a source peer are not reciprocals of each other [19] due to Jensen's inequality (see [32], p. 579), algorithms $\mathbb{PD}_1$ and $\mathbb{PD}_2$ are different.

Fig. 1 A parallel download and chunk allocation algorithm with probing

Algorithm $\mathbb{PD}_3$: Parallel Download and Chunk Allocation with Probing
Input: A file of size S and r source peers $1, 2, ..., r$.
Output: A download schedule for the file.

`for` $(i \leftarrow 1; i \le r; i{+}{+})$ `do in parallel` (1)
    download chunk i of size S^* from source peer i; (2)
`end do;` (3)
`for` $(i \leftarrow 1; i \le r; i{+}{+})$ `do in parallel` (4)
    set the size of chunk i to be (5)

$$S_i \leftarrow \left(\frac{C_i}{C_1 + C_2 + \cdots + C_r}\right)(S - rS^*);$$

`end do;` (6)
`for` $(i \leftarrow 1; i \le r; i{+}{+})$ `do in parallel` (7)
    download chunk i of size S_i from source peer i; (8)
`end do.` (9)

3.4.2 Analysis

The following theorem gives the expected parallel download time of algorithm $\mathbb{PD}_2$ for a file of size S from r source peers.

Theorem 5 *The expected parallel download time of algorithm $\mathbb{PD}_2$ is*

$$\boldsymbol{E}(T_{\mathbb{PD}_2}(S, r)) = S\boldsymbol{E}(T_{\mathbb{PD}_2}(1, r)),$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_2}(1, r)) = \left(\frac{1}{\boldsymbol{E}(C_1) + \boldsymbol{E}(C_2) + \cdots + \boldsymbol{E}(C_r)}\right) \times \sum_{i=1}^{r} \boldsymbol{E}(C_i) \int_0^\infty \frac{f_{C_i}(c)}{c} \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(C_{i'})}{\boldsymbol{E}(C_i)} c\right)\right) dc,$$

is the expected download time of one unit of data.

Proof Similar to the analysis of algorithm $\mathbb{PD}_1$, the expected parallel download time of algorithm $\mathbb{PD}_2$ can be obtained as follows,

$$\begin{aligned}
&\boldsymbol{E}(T_{\mathbb{PD}_2}(S, r)) \\
&\quad = \sum_{i=1}^{r} \int_0^\infty \frac{S_i}{c} f_{C_i}(c) \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{S_{i'}}{S_i} c\right)\right) dc \\
&\quad = \sum_{i=1}^{r} \int_0^\infty \frac{S_i}{c} f_{C_i}(c) \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(C_{i'})}{\boldsymbol{E}(C_i)} c\right)\right) dc \\
&\quad = \sum_{i=1}^{r} \int_0^\infty \frac{S}{c} \left(\frac{\boldsymbol{E}(C_i)}{\boldsymbol{E}(C_1) + \boldsymbol{E}(C_2) + \cdots + \boldsymbol{E}(C_r)}\right) \\
&\qquad \times f_{C_i}(c) \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(C_{i'})}{\boldsymbol{E}(C_i)} c\right)\right) dc \\
&\quad = S\boldsymbol{E}(T_{\mathbb{PD}_2}(1, r)),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_2}(1, r)) = \left(\frac{1}{\boldsymbol{E}(C_1) + \boldsymbol{E}(C_2) + \cdots + \boldsymbol{E}(C_r)}\right) \times \sum_{i=1}^{r} \boldsymbol{E}(C_i) \int_0^\infty \frac{f_{C_i}(c)}{c} \prod_{i' \ne i} \left(1 - F_{C_{i'}}\left(\frac{\boldsymbol{E}(C_{i'})}{\boldsymbol{E}(C_i)} c\right)\right) dc.$$

This proves the theorem. □

For r source peers with $\boldsymbol{E}(C_1) = \boldsymbol{E}(C_2) = \cdots = \boldsymbol{E}(C_r)$ (including the case of homogeneous source peers), algorithm $\mathbb{PD}_2$ works in exactly the same way as algorithm $\mathbb{PD}_0$, namely, dividing S into r chunks of equal size S/r.

3.5 Algorithm $\mathbb{PD}_3$ and analysis

3.5.1 Algorithm

Our algorithm $\mathbb{PD}_3$ for parallel download and chunk allocation with probing is given in Fig. 1. The algorithm consists of two stages. In the first stage (lines (1)–(3)), the service capacities of the r source peers are probed simultaneously by downloading one chunk of size S^* from each source, where S^* is a network-wide parameter agreed by and acceptable to all source and user peers. S^* should be reasonably chosen, e.g., 10 MB, to probe the current service capacities. After the first stage is performed, the algorithm knows the current service capacity C_i of each source peer i, where $1 \le i \le r$. The parallel download time of the first stage is

$$\max\{T_1(S^*), T_2(S^*), \ldots, T_r(S^*)\} = T_{\mathbb{PD}_0}(rS^*, r).$$

In the second stage (lines (4)–(9)), the rest of the file of size $S - rS^*$ is divided into r chunks of different sizes according to the C_i's detected in the first stage, such that all the r parallel downloads complete at the same time. It is clear that the size S_i of the ith chunk to be downloaded from source peer i should be

$$S_i = \left(\frac{C_i}{C_1 + C_2 + \cdots + C_r}\right)(S - rS^*),$$

for all $1 \le i \le r$. The parallel download time of the second stage is

$$T'_{\mathbb{PD}_3}(S, r) = \frac{S - rS^*}{C_1 + C_2 + \cdots + C_r}.$$

Summarizing the above discussion, the overall parallel download time of algorithm $\mathbb{PD}_3$ for a file of size S from r source peers is

$$T_{\mathbb{PD}_3}(S, r) = T_{\mathbb{PD}_0}(rS^*, r) + T'_{\mathbb{PD}_3}(S, r).$$

If service capacities do not change after probing, algorithm $\mathbb{PD}_3$ is already close to the optimal by achieving perfect load balancing in the second stage.

3.5.2 Analysis

Further analysis of $T'_{\mathbb{PD}_3}(S, r)$ depends on the $f_{C_i}(c)$'s. For instance, consider the case when C_i is a normal random variable with mean μ_{C_i} and variance $\sigma^2_{C_i}$, i.e.,

$$f_{C_i}(c) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-(c-\mu_{C_i})^2/(2\sigma^2_{C_i})},$$

where we assume that μ_{C_i} is reasonably large while σ_{C_i} is reasonably small such that the distribution of $f_{C_i}(c)$ in $(-\infty, 0]$ is negligible. It is well known that for a normal random variable C_i, the probability that its value is in the range $[\mu_{C_i} - 3\sigma_{C_i}, \mu_{C_i} + 3\sigma_{C_i}]$ is 99.73 %. The normal distribution is chosen here for analytical tractability. For all other real or synthetic probability distributions, simulations can be conducted for experimental performance evaluation (see Sect. 5.2.3). Let $\Phi(x)$ be the cdf of a standard normal distribution.

The following theorem gives the expected parallel download time of algorithm $\mathbb{PD}_3$ for a file of size S from r source peers.

Theorem 6 *If C_i is a normal random variable with parameters μ_{C_i} and $\sigma^2_{C_i}$, where $1 \le i \le r$, the expected parallel download time of algorithm $\mathbb{PD}_3$ is*

$$\boldsymbol{E}(T_{\mathbb{PD}_3}(S, r)) = S\boldsymbol{E}(T_{\mathbb{PD}_3}(1, r)) + K_{\mathbb{PD}_3}(r),$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_3}(1, r)) = \boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)),$$

and

$$K_{\mathbb{PD}_3}(r) = rS^*(\boldsymbol{E}(T_{\mathbb{PD}_0}(1, r)) - \boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r))),$$

and

$$\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)) = \sigma_r \int_{-\mu_r/\sigma_r}^{\infty} \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx,$$

with

$$\mu_r = \mu_{C_1} + \mu_{C_2} + \cdots + \mu_{C_r},$$

and

$$\sigma_r^2 = \sigma^2_{C_1} + \sigma^2_{C_2} + \cdots + \sigma^2_{C_r}.$$

Proof It is clear that if C_i is a normal random variable with parameters μ_{C_i} and $\sigma^2_{C_i}$, for all $1 \le i \le r$, then, $C_1 + C_2 + \cdots + C_r$ is also a normal random variable with parameters

$$\mu_r = \mu_{C_1} + \mu_{C_2} + \cdots + \mu_{C_r},$$

and

$$\sigma_r^2 = \sigma^2_{C_1} + \sigma^2_{C_2} + \cdots + \sigma^2_{C_r}.$$

This gives rise to the cdf of $T'_{\mathbb{PD}_3}(S, r)$ as follows,

$$\begin{aligned} F_{T'_{\mathbb{PD}_3}(S,r)}(t) &= \boldsymbol{P}\left[T'_{\mathbb{PD}_3}(S, r) \le t\right] \\ &= \boldsymbol{P}\left[\frac{S - rS^*}{C_1 + C_2 + \cdots + C_r} \le t\right] \\ &= \boldsymbol{P}\left[C_1 + C_2 + \cdots + C_r \ge \frac{S - rS^*}{t}\right] \\ &= 1 - \Phi\left(\frac{(S - rS^*)/t - \mu_r}{\sigma_r}\right), \end{aligned}$$

and the pdf of $T'_{\mathbb{PD}_3}(S, r)$ as follows,

$$f_{T'_{\mathbb{PD}_3}(S,r)}(t) = \frac{S - rS^*}{\sqrt{2\pi}\sigma_r t^2} e^{-((S-rS^*)/t-\mu_r)^2/(2\sigma_r^2)},$$

for all $t > 0$. The expectation of $T'_{\mathbb{PD}_3}(S, r)$ is

$$\begin{aligned}
&\boldsymbol{E}(T'_{\mathbb{PD}_3}(S, r)) \\
&= \int_0^\infty \left(1 - F_{T'_{\mathbb{PD}_3}(S,r)}(t)\right) dt \\
&= \int_0^\infty \Phi\left(\frac{(S - rS^*)/t - \mu_r}{\sigma_r}\right) dt \\
&= \int_\infty^{-\mu_r/\sigma_r} \Phi(x)\left(-\frac{\sigma_r(S - rS^*)}{(\sigma_r x + \mu_r)^2}\right) dx \\
&\qquad \left(\text{by letting } x = \frac{1}{\sigma_r}\left(\frac{S - rS^*}{t} - \mu_r\right)\right) \\
&= \sigma_r(S - rS^*)\int_{-\mu_r/\sigma_r}^\infty \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx \\
&= (S - rS^*)\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)),
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)) &= \sigma_r \int_{-\mu_r/\sigma_r}^\infty \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx \\
&= \sigma_r\left(\int_{-\mu_r/\sigma_r}^{3.5} \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx\right) \\
&\approx \sigma_r\left(\int_{-\mu_r/\sigma_r}^{3.5} \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{1}{(\sigma_r x + \mu_r)^2} dx\right) \\
&\qquad \text{(notice that } \Phi(x) \approx 1 \text{ for } x \ge 3.5) \\
&= \sigma_r\left(\int_{-\mu_r/\sigma_r}^{3.5} \frac{\Phi(x)}{(\sigma_r x + \mu_r)^2} dx \right.\\
&\qquad \left. + \frac{1}{\sigma_r(3.5\sigma_r + \mu_r)}\right).
\end{aligned}$$

Finally, we notice that the expectation of $T_{\mathbb{PD}_3}(S, r)$ is simply

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_3}(S, r)) &= \boldsymbol{E}(T_{\mathbb{PD}_0}(rS^*, r)) + \boldsymbol{E}(T'_{\mathbb{PD}_3}(S, r)) \\
&= rS^*\boldsymbol{E}(T_{\mathbb{PD}_0}(1, r)) \\
&\quad + (S - rS^*)\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)) \\
&= S\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)) \\
&\quad + rS^*(\boldsymbol{E}(T_{\mathbb{PD}_0}(1, r)) - \boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r))) \\
&= S\boldsymbol{E}(T_{\mathbb{PD}_3}(1, r)) + K_{\mathbb{PD}_3}(r),
\end{aligned}$$

where

$$\boldsymbol{E}(T_{\mathbb{PD}_3}(1, r)) = \boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r))$$

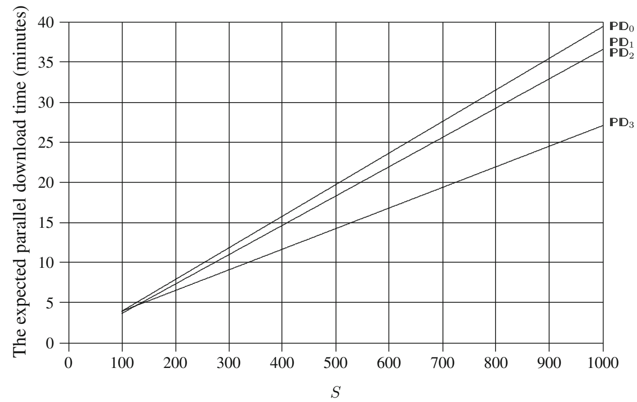


Fig. 2 The expected parallel download time versus file size

and

$$K_{\mathbb{PD}_3}(r) = rS^*(\boldsymbol{E}(T_{\mathbb{PD}_0}(1, r)) - \boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)))$$

are constants independent of S. □

We also notice that the pdf of $T_{\mathbb{PD}_3}(S, r)$, which involves the pdf of $T_{\mathbb{PD}_0}(rS^*, r) + T'_{\mathbb{PD}_3}(S, r)$, is very complicated.

For homogeneous source peers, we have $\mu_r = r\mu_C$, and $\sigma_r^2 = r\sigma_C^2$. Consequently,

$$\boldsymbol{E}(T'_{\mathbb{PD}_3}(1, r)) = \frac{\sigma_C}{\sqrt{r}} \int_{-\sqrt{r}\mu_C/\sigma_C}^\infty \frac{\Phi(x)}{(\sigma_C x + \sqrt{r}\mu_C)^2} dx.$$

4 Performance comparison

In this section, we present a numerical example to compare the performance of the algorithms discussed in this paper. As in most P2P file sharing and exchange systems, the file sizes S are in the range $10 - 1500$ MB [1]. We set the chunk size $S^* = 10$ MB. The service capacity of a source peer is in the range $50 - 1000$ kbps, i.e., $0.375 - 7.5$ MB/min.

Let us consider a P2P file sharing system with $n = 10$ source peers. Assume that C_i has a normal distribution with parameters μ_{C_i} and $\sigma_{C_i}^2$, where $\mu_{C_i} = 3 + 0.2(i - 1)$ and $\sigma_{C_i} = 0.5 + 0.05(i - 1)$, for all $1 \le i \le n$. (Notice that a larger service capacity tends to have greater variance. Also, these values are for demonstration purpose only.) In Fig. 2, we show the expected parallel download time of algorithm $\mathbb{PD}_\ell$ as a function of file size S, where $0 \le \ell \le 3$, and $100 \le S \le 1000$. All the data are calculated by using Theorems 3–6. It is observed that algorithms $\mathbb{PD}_1$ and $\mathbb{PD}_2$ perform better than algorithm $\mathbb{PD}_0$ by careful chunk allocation based on the expected behavior of source peers. Although algorithm $\mathbb{PD}_2$ performs slightly better than algorithm $\mathbb{PD}_1$, the difference is not noticeable in this example. Algorithm $\mathbb{PD}_3$ (with $S^* =$

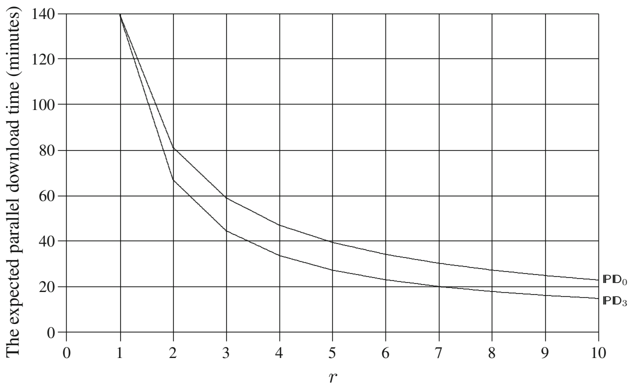


Fig. 3 The expected parallel download time versus parallelism ($S = 500$)

10) performs significantly better than $\mathbb{PD}_0$, $\mathbb{PD}_1$, and $\mathbb{PD}_2$ by probing source peers.

In Fig. 3, we consider a P2P file sharing system with r homogeneous source peers whose service capacity has a normal distribution with parameters $\mu_C = 3.9$ and $\sigma_C^2 = 1.0$. We show the expected parallel download time of algorithm $\mathbb{PD}_\ell(\ell = 0, 3)$ as a function of the parallelism r, where $1 \le r \le 10$, for a file of size $S = 500$. Since algorithms $\mathbb{PD}_0$, $\mathbb{PD}_1$, $\mathbb{PD}_2$ are identical for homogeneous source peers, we only show the curves for $\mathbb{PD}_0$ and $\mathbb{PD}_3$. It is observed that the the expected parallel download time decreases dramatically, although not linearly, as the number r of source peers increases. This implies that parallelism indeed improves the performance of file downloading.

5 Multiple downloads and optimal parallelism

For multiple downloads, we have N files of sizes $S_1, S_2, \ldots, S_N$ to be downloaded from n source peers $1, 2, \ldots, n$. All source peers in this section are homogeneous whose service capacities are random variable C.

5.1 Algorithm $\mathbb{PD}_\ell^*$ and analysis

5.1.1 Algorithm

Our algorithm $\mathbb{PD}_\ell^*$ for multiple downloads works as follows. We divide the n source peers into $k = n/r$ clusters, each having r source peers. We also divide the N files into k groups, where group j contains $b = N/k = Nr/n$ files $S_{(j-1)b+1}, S_{(j-1)b+2}, \ldots, S_{jb}$, for all $1 \le j \le k$. For examples, group 1 contains $S_1, S_2, \ldots, S_b$; group 2 contains $S_{b+1}, S_{b+2}, \ldots, S_{2b}$; and so on. Let

$$G_j = S_{(j-1)b+1} + S_{(j-1)b+2} + \cdot s + S_{jb}$$

be the total workload of group j. A file in group j is downloaded from the r source peers in cluster j in parallel by using the algorithm $\mathbb{PD}_\ell$, where $0 \le \ell \le 3$. Files in the same group are downloaded one at a time.

5.1.2 Analysis

Let $T_{\mathbb{PD}_\ell}(G_j, r)$ denote the parallel download time of algorithm $\mathbb{PD}_\ell$ for all files in group j from the r source peers in cluster j, where $1 \le j \le k$, and $0 \le \ell \le 3$. It is clear that

$$T_{\mathbb{PD}_\ell}(G_j, r) = T_{\mathbb{PD}_\ell}(S_{(j-1)b+1}, r) + T_{\mathbb{PD}_\ell}(S_{(j-1)b+2}, r) + \cdots + T_{\mathbb{PD}_\ell}(S_{jb}, r).$$

Our analysis in the last section reveals that for all $0 \le \ell \le 3$, the expected parallel download time of algorithm $\mathbb{PD}_\ell$ for a file of size S from r source peers is a linear function of S, namely,

$$\boldsymbol{E}(T_{\mathbb{PD}_\ell}(S, r)) = S\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + K_{\mathbb{PD}_\ell}(r).$$

In fact, $K_{\mathbb{PD}_\ell}(r) = 0$ except for $\ell = 3$. Hence, the expected parallel download time of algorithm $\mathbb{PD}_\ell$ for all files in group j from the r source peers in cluster j is a linear function of G_j calculated as follows,

$$\begin{aligned}
&\boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_j, r)) \\
&\quad = \boldsymbol{E}(T_{\mathbb{PD}_\ell}(S_{(j-1)b+1}, r)) + \boldsymbol{E}(T_{\mathbb{PD}_\ell}(S_{(j-1)b+2}, r)) \\
&\qquad + \cdots + \boldsymbol{E}(T_{\mathbb{PD}_\ell}(S_{jb}, r)) \\
&\quad = (S_{(j-1)b+1}\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + K_{\mathbb{PD}_\ell}(r)) \\
&\qquad + (S_{(j-1)b+2}\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + K_{\mathbb{PD}_\ell}(r)) \\
&\qquad + \cdots + (S_{jb}\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + K_{\mathbb{PD}_\ell}(r)) \\
&\quad = (S_{(j-1)b+1} + S_{(j-1)b+2} + \cdots + S_{jb})\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) \\
&\qquad + bK_{\mathbb{PD}_\ell}(r) \\
&\quad = G_j\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + bK_{\mathbb{PD}_\ell}(r).
\end{aligned}$$

Let $T_{\mathbb{PD}_\ell^*}(N, n)$ denote the download time of algorithm $\mathbb{PD}_\ell^*$ for N files from n source peers. Then, we have

$$\begin{aligned}
&T_{\mathbb{PD}_\ell^*}(N, n) = \max\{T_{\mathbb{PD}_\ell}(G_1, r), \\
&\quad T_{\mathbb{PD}_\ell}(G_2, r), \ldots, T_{\mathbb{PD}_\ell}(G_k, r)\}.
\end{aligned}$$

Unfortunately, the pdf of $T_{\mathbb{PD}_\ell^*}(N, n)$ (in fact, even the pdf of $T_{\mathbb{PD}_\ell}(G_j, r)$) is too complicated, which makes the evaluation of $\boldsymbol{E}(T_{\mathbb{PD}_\ell^*}(N, n))$ analytically intractable. Instead, we consider

$$\begin{aligned}
&T'_{\mathbb{PD}_\ell^*}(N, n) = \max\{\boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_1, r)), \\
&\quad \boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_2, r)), \ldots, \boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_k, r))\},
\end{aligned}$$

where the expectation of $T_{\mathbb{PD}_\ell}(G_j, r)$ is with respect to the random service capacities of the r source peers in cluster j. Notice that $T'_{\mathbb{PD}^*_\ell}(N, n)$ is also a random variable, whose expectation with respect to the random G_j's is given by the following theorem. (Notice that $T_{\mathbb{PD}^*_\ell}(N, n)$ and $T'_{\mathbb{PD}^*_\ell}(N, n)$ are different. While $T'_{\mathbb{PD}^*_\ell}(N, n)$ may reveal some useful information, it cannot substitute $T_{\mathbb{PD}^*_\ell}(N, n)$. To understand $T_{\mathbb{PD}^*_\ell}(N, n)$, we can conduct experiments to study $\boldsymbol{E}(T_{\mathbb{PD}^*_\ell}(N, n))$. Such experiments are reported in Sect. 5.2.3.)

Theorem 7 *If $S_1, S_2, \ldots, S_N$ are independent and identically distributed normal random variables with parameters μ_S and σ_S^2, we have*

$$\boldsymbol{E}(T'_{\mathbb{PD}^*_\ell}(N, n)) = \boldsymbol{E}(W)\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + \frac{Nr}{n} K_{\mathbb{PD}_\ell}(r),$$

where

$$W = \max\{G_1, G_2, \ldots, G_k\},$$

and

$$\boldsymbol{E}(W) = \sqrt{\frac{Nr}{n}}\sigma_S\left(\sqrt{\frac{Nr}{n}} \cdot \frac{\mu_S}{\sigma_S} - 3.5 + \int_{-3.5}^{\infty}\left(1 - (\Phi(x))^k\right)dx\right).$$

Proof It is clear that

$$\begin{aligned}
&T'_{\mathbb{PD}^*_\ell}(N, n)\\
&\quad = \max\{\boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_1, r)),\\
&\qquad \boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_2, r)), \ldots, \boldsymbol{E}(T_{\mathbb{PD}_\ell}(G_k, r))\}\\
&\quad = \max\{G_1\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)),\\
&\qquad G_2\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)), \ldots, G_k\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r))\}\\
&\qquad + bK_{\mathbb{PD}_\ell}(r)\\
&\quad = \max\{G_1, G_2, \ldots, G_k\}\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r))\\
&\qquad + bK_{\mathbb{PD}_\ell}(r)\\
&\quad = W\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + bK_{\mathbb{PD}_\ell}(r),
\end{aligned}$$

where

$$W = \max\{G_1, G_2, \ldots, G_k\}.$$

Notice that

$$\boldsymbol{E}(T'_{\mathbb{PD}^*_\ell}(N, n)) = \boldsymbol{E}(W)\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r)) + \frac{Nr}{n} K_{\mathbb{PD}_\ell}(r),$$

where the terms $\boldsymbol{E}(W)$ and $b = Nr/n$ include the effect of the overhead of intercluster parallelism and the terms $\boldsymbol{E}(T_{\mathbb{PD}_\ell}(1, r))$ and $K_{\mathbb{PD}_\ell}(r)$ include the effect of the overhead of intracluster parallelism.

To evaluate $\boldsymbol{E}(W)$, we need further information of $S_1, S_2, \ldots, S_N$. If $S_1, S_2, \ldots, S_N$ are independent and identically distributed random variables with pdf $f_S(s)$ in $[0, \infty)$, and furthermore, $f_S(s)$ has a normal distribution with parameters μ_S and σ_S^2, then, G_j is a normal random variable with parameters $\mu_{G_j} = b\mu_S$ and $\sigma_{G_j}^2 = b\sigma_S^2$, for all $1 \le j \le k$. This gives the cdf of W as

$$\begin{aligned}
F_W(w) &= \boldsymbol{P}[W \le w]\\
&= \prod_{j=1}^{k} \boldsymbol{P}[G_j \le w]\\
&= \prod_{j=1}^{k} F_{G_j}(w)\\
&= \prod_{j=1}^{k} \Phi\left(\frac{w - \mu_{G_j}}{\sigma_{G_j}}\right)\\
&= \left(\Phi\left(\frac{w - b\mu_S}{\sqrt{b}\sigma_S}\right)\right)^k,
\end{aligned}$$

and the pdf of W as

$$f_W(w) = k\left(\Phi\left(\frac{w - b\mu_S}{\sqrt{b}\sigma_S}\right)\right)^{k-1}\frac{1}{\sqrt{2\pi b}\sigma_S}e^{-(w - b\mu_S)^2/(2b\sigma_S^2)},$$

for all $w > 0$, and the expectation of W as

$$\begin{aligned}
\boldsymbol{E}(W) &= \int_0^\infty (1 - F_W(w))\,dw\\
&= \int_0^\infty \left(1 - \left(\Phi\left(\frac{w - b\mu_S}{\sqrt{b}\sigma_S}\right)\right)^k\right)dw\\
&= \int_{-\sqrt{b}\mu_S/\sigma_S}^{\infty} \sqrt{b}\sigma_S\left(1 - (\Phi(x))^k\right)dx\\
&\qquad \left(\text{by letting } x = \frac{w - b\mu_S}{\sqrt{b}\sigma_S}\right)\\
&= \sqrt{b}\sigma_S\left(\int_{-\sqrt{b}\mu_S/\sigma_S}^{-3.5}\left(1 - (\Phi(x))^k\right)dx\right.\\
&\qquad \left. + \int_{-3.5}^{\infty}\left(1 - (\Phi(x))^k\right)dx\right)\\
&\approx \sqrt{b}\sigma_S\left(\sqrt{b}\mu_S/\sigma_S - 3.5\right.\\
&\qquad \left. + \int_{-3.5}^{\infty}\left(1 - (\Phi(x))^k\right)dx\right)\\
&= \sqrt{\frac{Nr}{n}}\sigma_S\left(\sqrt{\frac{Nr}{n}} \cdot \frac{\mu_S}{\sigma_S} - 3.5\right.\\
&\qquad \left. + \int_{-3.5}^{\infty}\left(1 - (\Phi(x))^k\right)dx\right),
\end{aligned}$$

where we notice that $\Phi(x) \approx 0$ for $x \leq -3.5$. □

We would like to mention that if $S_1, S_2, \ldots, S_N$ are not normal random variables and have any real or synthetic probability distributions, simulations can be conducted for experimental performance study (see Sect. 5.2.3).

5.2 Optimal parallelism

5.2.1 Overhead of parallelism

There are two kinds of overhead of parallelism, i.e., synchronization for parallel download.

- Intracluster overhead – This is the peer level overhead, i.e., the extra time for synchronization of the r parallel downloads of a file caused by load imbalance among chunks and/or heterogeneity of service capacities. If a file of size S_a is divided into r chunks of sizes $S_{a,1}, S_{a,2}, \ldots, S_{a,r}$, and chunk $S_{a,i}$ is downloaded from source peer i, for all $1 \leq i \leq r$ in parallel, it is practically impossible to achieve $T_1(S_{a,1}) = T_2(S_{a,2}) = \cdots = T_r(S_{a,r})$.
- Intercluster overhead – This is the cluster level overhead, i.e., the extra time for synchronization of the k parallel downloads of k groups of files caused by load imbalance among groups and/or heterogeneity of service capacities. It is practically impossible to have the k clusters to complete their downloads at the same time.

The parameter r is the degree of parallelism at the peer level. The parameter k is the degree of parallelism at the cluster level. When r is small, the intracluster overhead of parallelism is small; however, the number of clusters k is large, which causes large load imbalance among the clusters and the intercluster overhead of parallelism is large. On the other hand, when r is large, the intracluster overhead of parallelism is large due to increased load imbalance among the source peers in a cluster, while the intercluster overhead of parallelism is small due to decreased load imbalance among the clusters. Therefore, there is an optimal choice of the parallelism r which minimizes the combined effect of intracluster and intercluster overhead of parallel download and load imbalance.

5.2.2 Numerical data

Our main problem in this section is to find r such that $\boldsymbol{E}(T'_{\mathbb{PD}^*_\ell}(N, n))$ is minimized.

Consider the case when C is a normal random variable with parameters μ_C and σ_C^2. Based on Theorems 3,4, and 5 of the last section, we have

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) &= \boldsymbol{E}(T_{\mathbb{PD}_1}(1,r)) \\
&= \boldsymbol{E}(T_{\mathbb{PD}_2}(1,r)) \\
&= \frac{1}{r}\int_0^\infty \frac{1}{c^2}\left(1-(1-F_C(c))^r\right)dc \\
&= \frac{1}{r}\int_0^\infty \frac{1}{c^2}\left(1-\left(1-\Phi\left(\frac{c-\mu_C}{\sigma_C}\right)\right)^r\right)dc \\
&= \frac{\sigma_C}{r}\int_{-\mu_C/\sigma_C}^\infty \frac{\left(1-(1-\Phi(x))^r\right)}{(\sigma_C x+\mu_C)^2}dx \\
&\qquad \left(\text{by letting } x = \frac{c-\mu_C}{\sigma_C}\right) \\
&= \frac{\sigma_C}{r}\left(\int_{-\mu_C/\sigma_C}^{3.5} \frac{\left(1-(1-\Phi(x))^r\right)}{(\sigma_C x+\mu_C)^2}dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{\left(1-(1-\Phi(x))^r\right)}{(\sigma_C x+\mu_C)^2}dx\right) \\
&\approx \frac{\sigma_C}{r}\left(\int_{-\mu_C/\sigma_C}^{3.5} \frac{\left(1-(1-\Phi(x))^r\right)}{(\sigma_C x+\mu_C)^2}dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{1}{(\sigma_C x+\mu_C)^2}dx\right) \\
&= \frac{\sigma_C}{r}\left(\int_{-\mu_C/\sigma_C}^{3.5} \frac{\left(1-(1-\Phi(x))^r\right)}{(\sigma_C x+\mu_C)^2}dx \right.\\
&\qquad \left. + \frac{1}{\sigma_C(3.5\sigma_C+\mu_C)}\right),
\end{aligned}$$

where we notice that $\Phi(x) \approx 1$ for $x \geq 3.5$. By Theorem 6, we also have

$$\begin{aligned}
\boldsymbol{E}(T_{\mathbb{PD}_3}(1,r)) &= \frac{\sigma_C}{\sqrt{r}}\int_{-\sqrt{r}\mu_C/\sigma_C}^\infty \frac{\Phi(x)}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx \\
&= \frac{\sigma_C}{\sqrt{r}}\left(\int_{-\sqrt{r}\mu_C/\sigma_C}^{3.5} \frac{\Phi(x)}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{\Phi(x)}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx\right) \\
&\approx \frac{\sigma_C}{\sqrt{r}}\left(\int_{-\sqrt{r}\mu_C/\sigma_C}^{3.5} \frac{\Phi(x)}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx \right.\\
&\qquad \left. + \int_{3.5}^\infty \frac{1}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx\right) \\
&= \frac{\sigma_C}{\sqrt{r}}\left(\int_{-\sqrt{r}\mu_C/\sigma_C}^{3.5} \frac{\Phi(x)}{(\sigma_C x+\sqrt{r}\mu_C)^2}dx \right.\\
&\qquad \left. + \frac{1}{\sigma_C(3.5\sigma_C+\sqrt{r}\mu_C)}\right),
\end{aligned}$$

and

$$K_{\mathbb{PD}_3}(r) = rS^*(\boldsymbol{E}(T_{\mathbb{PD}_0}(1,r)) - \boldsymbol{E}(T_{\mathbb{PD}_3}(1,r))).$$

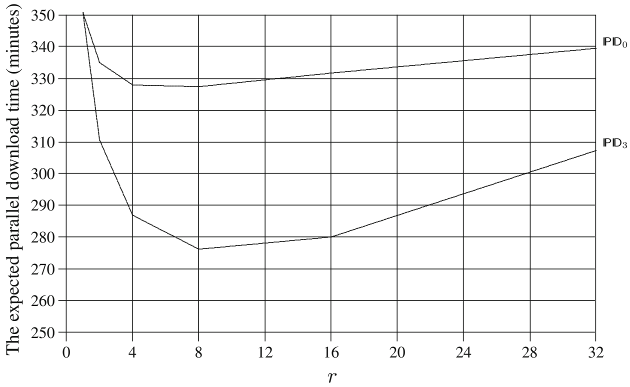


Fig. 4 The expected parallel download time versus parallelism ($n = 32$, $N = 64$)

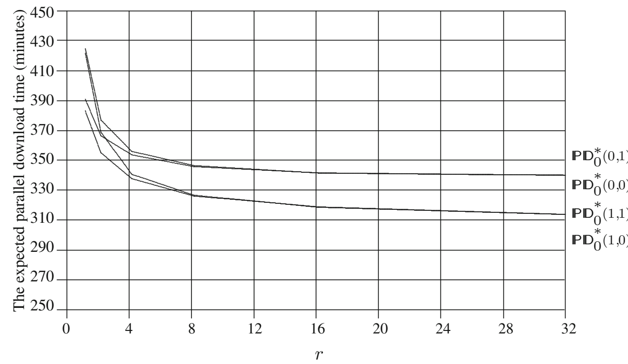


Fig. 5 The expected parallel download time $\boldsymbol{E}(T_{\mathbb{PD}_0^*}(N, n))$ versus parallelism ($n = 32$, $N = 64$)

Let us consider a P2P file sharing system with $n = 32$ homogeneous source peers whose service capacity has a normal distribution with parameters $\mu_C = 4.0$ and $\sigma_C = 0.5$. Assume that we are going to download $N = 64$ files whose sizes have a normal distribution with parameters $\mu_S = 500$ and $\sigma_S = 130$. In Fig. 4, we show the expected parallel download time of algorithm $\mathbb{PD}_\ell(\ell = 0, 3)$ as a function of the parallelism r, where $1 \le r \le 32$. All the data are calculated by using Theorem 7. Since algorithms $\mathbb{PD}_0, \mathbb{PD}_1, \mathbb{PD}_2$ are identical for homogeneous source peers, we only show the curves for $\mathbb{PD}_0$ and $\mathbb{PD}_3$ (with $S^* = 10$). It is observed that the expected parallel download time is sensitive to the configuration of a P2P file sharing system, i.e., how the source peers are divided into clusters. Assume that $r = 1, 2, 4, 8, 16, 32$. Then, the best parallelism for both algorithms $\mathbb{PD}_0$ and $\mathbb{PD}_3$ is $r = 8$.

5.2.3 Simulation results

As mentioned before, $T_{\mathbb{PD}_\ell^*}(N, n)$, the download time of algorithm $\mathbb{PD}_\ell^*$ for N files from n source peers, is an extremely sophisticated random variable beyond analysis. However, simulations can be performed to evaluate $\boldsymbol{E}(T_{\mathbb{PD}_\ell^*}(N, n))$, where the expectation is taken with respect to the random service capacities $C_1, C_2, \ldots, C_n$ and random file sizes $S_1, S_2, \ldots, S_N$.

Again, we consider parallel downloading of $N = 64$ files from a P2P file sharing system with $n = 32$ homogeneous source peers. We consider algorithm $\mathbb{PD}_\ell^*(p, q)(\ell = 0, 3)$. If $p = 0$, the service capacities have a normal distribution with parameters $\mu_C = 4.0$ and $\sigma_C = 0.5$. If $p = 1$, the service capacities have a uniform distribution in the range $[4 - \sqrt{3}/2, 4 + \sqrt{3}/2]$, with mean $\mu_C = 4.0$ and variance $\sigma_C^2 = 0.25$ (i.e., $\sigma_C = 0.5$). If $q = 0$, the file sizes have a normal distribution with parameters $\mu_S = 500$ and $\sigma_S = 130$. If $q = 1$, the file sizes have a Pareto distribution (commonly used for modeling a power law probability distribution) with pdf

$$f_S(s) = \frac{\alpha\beta^\alpha}{s^{\alpha+1}},$$

in the range $[\beta, \infty)$, where α is the shape parameter and β is the scale parameter. We set $\alpha = 5$ and $\beta = 400$, such that

$$\mu_S = \frac{\alpha\beta}{\alpha - 1} = 500,$$

and

$$\sigma_S = \frac{\beta}{\alpha - 1}\sqrt{\frac{\alpha}{\alpha - 2}} = 100\sqrt{\frac{5}{3}} \approx 129.$$

Since algorithms $\mathbb{PD}_0, \mathbb{PD}_1, \mathbb{PD}_2$ are identical for homogeneous source peers, we only consider $\mathbb{PD}_0$ and $\mathbb{PD}_3$.

In Fig. 5, we show the expected parallel download time $\boldsymbol{E}(T_{\mathbb{PD}_0^*}(N, n))$ with $\mathbb{PD}_0^*(0, 0)$, $\mathbb{PD}_0^*(0, 1)$, $\mathbb{PD}_0^*(1, 0)$, $\mathbb{PD}_0^*(1, 1)$. All the data are obtained from simulations. In each experiment, we generate n random service capacities and N random file sizes based on the probability distributions specified by p and q. Then algorithm $\mathbb{PD}_0^*$ is used to get the parallel download time. Consider the jth cluster of source peers $C_{(j-1)r+1}, C_{(j-1)r+2}, \ldots, C_{jr}$ which provide files $S_{(j-1)b+1}, S_{(j-1)b+2}, \ldots, S_{jb}$ in group j. Then, we have

$$\begin{aligned} T_{\mathbb{PD}_0}(G_j, r) &= \sum_{i=1}^{b} \max\left(\frac{S_{(j-1)b+i}}{rC_{(j-1)r+1}}, \frac{S_{(j-1)b+i}}{rC_{(j-1)r+2}}, \ldots, \frac{S_{(j-1)b+i}}{rC_{jr}}\right) \\ &= \sum_{i=1}^{b} \frac{S_{(j-1)b+i}}{r} \cdot \frac{1}{\min(C_{(j-1)r+1}, C_{(j-1)r+2}, \ldots, C_{jr})} \\ &= \frac{1}{r} \cdot \frac{1}{\min(C_{(j-1)r+1}, C_{(j-1)r+2}, \ldots, C_{jr})} \sum_{i=1}^{b} S_{(j-1)b+i} \\ &= \frac{G_j}{r} \cdot \frac{1}{\min(C_{(j-1)r+1}, C_{(j-1)r+2}, \ldots, C_{jr})}, \end{aligned}$$

for all $1 \le j \le k$, and

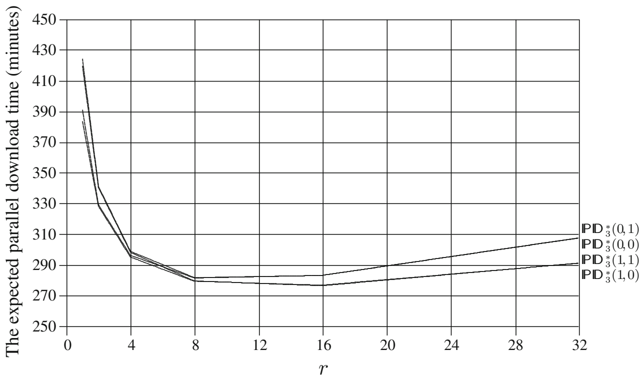


Fig. 6 The expected parallel download time $\boldsymbol{E}(T_{\mathbb{PD}_3^*}(N,n))$ versus parallelism ($n = 32$, $N = 64$)

$$T_{\mathbb{PD}_0^*}(N,n) = \max\{T_{\mathbb{PD}_0}(G_1,r), T_{\mathbb{PD}_0}(G_2,r), \ldots, T_{\mathbb{PD}_0}(G_k,r)\}.$$

Each data in the figure is the average of the data obtained from 10,000 repeated experiments, such that the 99 % confidence interval is no more than ±0.57 %. It is observed from these simulation results that $\boldsymbol{E}(T_{\mathbb{PD}_0^*}(N,n))$ is actually a decreasing function of r, i.e., larger parallelism with fewer clusters result in reduced parallel downloading time.

In Fig. 6, we show the expected parallel download time $\boldsymbol{E}(T_{\mathbb{PD}_3^*}(N,n))$ with $\mathbb{PD}_3^*(0,0)$, $\mathbb{PD}_3^*(0,1)$, $\mathbb{PD}_3^*(1,0)$, $\mathbb{PD}_3^*(1,1)$. We follow the same procedure as Fig. 5. It is noticed that

$$\begin{aligned}
T_{\mathbb{PD}_3}(G_j,r) &= \sum_{i=1}^{b}\left(\max\left(\frac{S^*}{C_{(j-1)r+1}},\ldots,\frac{S^*}{C_{jr}}\right) + \frac{S_{(j-1)b+i} - rS^*}{C_{(j-1)r+1}+\cdots+C_{jr}}\right)\\
&= \sum_{i=1}^{b}\left(\frac{S^*}{\min(C_{(j-1)r+1},\ldots,C_{jr})} + \frac{S_{(j-1)b+i} - rS^*}{C_{(j-1)r+1}+\cdots+C_{jr}}\right)\\
&= b\left(\frac{S^*}{\min(C_{(j-1)r+1},\ldots,C_{jr})} - \frac{rS^*}{C_{(j-1)r+1}+\cdots+C_{jr}}\right) + \frac{1}{C_{(j-1)r+1}+\cdots+C_{jr}}\sum_{i=1}^{b} S_{(j-1)b+i}\\
&= b\left(\frac{S^*}{\min(C_{(j-1)r+1},\ldots,C_{jr})} - \frac{rS^*}{C_{(j-1)r+1}+\cdots+C_{jr}}\right) + \frac{G_j}{C_{(j-1)r+1}+\cdots+C_{jr}},
\end{aligned}$$

for all $1 \le j \le k$, and

$$T_{\mathbb{PD}_3^*}(N,n) = \max\{T_{\mathbb{PD}_3}(G_1,r), T_{\mathbb{PD}_3}(G_2,r), \ldots, T_{\mathbb{PD}_3}(G_k,r)\}.$$

Each data in the figure is the average of the data obtained from 10,000 repeated experiments, such that the 99 % confidence interval is no more than ±0.56 %. It is observed from these simulation results that $\boldsymbol{E}(T_{\mathbb{PD}_3^*}(N,n))$ decreases as r increases, and beyond certain point, increases. When service capacities have a normal distribution, the optimal parallelism is $r = 8$. When service capacities have a uniform distribution, the optimal parallelism is $r = 16$.

From both Figs. 5 and 6, we notice that $\boldsymbol{E}(T_{\mathbb{PD}_\ell^*}(N,n)) > \boldsymbol{E}(T'_{\mathbb{PD}_\ell^*}(N,n))$. Furthermore, we observe that different distributions of service capacities and file sizes have noticeably different performance, even with the same mean and variance. In particular, for service capacities, the normal distribution has longer download time than the uniform distribution; and for file sizes, the Pareto distribution has longer download time than the normal distribution.

6 Conclusions

We have addressed the problem of efficient parallel file download in peer-to-peer networks with random service capacities. We gave a precise analysis of the expected download time when the service capacity of a source peer is a random variable. We developed a general framework for analyzing the expected download time of a parallel download and chunk allocation algorithm, and applied the framework to the analysis of several algorithms. We have proposed two chunk allocation algorithms for parallel download. We also proposed an algorithm of probing high-capacity peers and analyzed the expected download time of the algorithm. We compared the performance of these parallel file download algorithms in P2P networks with random service capacities. We also extended the above parallel download algorithms to multiple file download by dividing source peers into clusters and analyzed the expected download times. We pointed out that there is an important issue of optimal parallelism which minimizes the combined effect of intracluster and intercluster overhead of parallel download and load imbalance.

This paper has made some initial effort of analytical performance study of parallel download algorithms. Further research should take more challenging scenarios into consideration. For instance, for a hot and new eagerly awaited release, how to benefit from peers with incomplete files is very interesting and important for improved performance. Another direction worth of investigation is to consider multiple smaller chunks from each source peer, instead of one single larger chunk, where the sizes of smaller chunks can be dynamically adjusted according to the current service capacities (assuming that service capacities still change after probing and multiple probings are required) to increase the quality of load balancing. It is conceivable that analysis of such a sophisticated method is quite challenging.

Acknowledgments The author deeply appreciates the comments from three anonymous reviewers that have helped to improve the quality of the manuscript.

References

1. http://blogplots.blogspot.com/2008/02/p2p-file-size-distribution.html. Accessed 18 July 2016.
2. http://en.wikipedia.org/wiki/File_sharing. Accessed 18 July 2016.
3. http://en.wikipedia.org/wiki/Peer-to-peer. Accessed 18 July 2016.
4. Adler, M., Kumar, R., Ross, K., Rubenstein, D., Suel, T., & Yao, D. D. (2005). Optimal peer selection for P2P downloading and streaming. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies* (vol. 3, pp. 1538–1549).
5. Adler, M., Kumar, R., Ross, K., Rubenstein, D., Turner, D., & Yao, D. D. (2004) Optimal peer selection in a free-market peer-resource economy. In *Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems*.
6. Bernstein, D. S., Feng, Z., Levine, B. N., & Zilberstein, S. (2003). Adaptive peer selection. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*.
7. Byers, J. W., Luby, M., & Mitzenmacher, M. (1999). Accessing multiple mirror sites in parallel: Using Tornado codes to speed up downloads. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies* (vol. 1, pp. 275–283).
8. Carter, R. L., & Crovella, M. E. (1999). On the network impact of dynamic server selection. *Computer Networks*, *31*(23–24), 2529–2558.
9. Chiu, Y.-M. (2009). *On the performance of peer selection strategies in stochastic peer-to-peer networks*. Ph.D. dissertation, North Carolina State University, Electrical Engineering.
10. Chiu, Y.-M., & Eun, D. Y. (2008). Minimizing file download time in stochastic peer-to-peer networks. *IEEE/ACM Transactions on Networking*, *16*(2), 253–266.
11. Dykes, S. G., Robbins, K. A., & Jeffery, C. L. (2000). An empirical evaluation of client-side server selection algorithms. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies* (vol. 3, pp. 1361–1370).
12. Gaeta, R., Gribaudo, M., Manini, D., & Sereno, M. (2006). Analysis of resource transfers in peer-to-peer file sharing applications using fluid models. *Performance Evaluation*, *63*, 149–174.
13. Ge, Z., Figueiredo, D. R., Jaiswal, S., Kurose, J., & Towsley, D. (2003). Modeling peer-peer file sharing systems. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies* (vol. 3, pp. 2188–2198).
14. Gkantsidis, C., Ammar, M., & Zegura, E. (2003). On the effect of large-scale deployment of parallel downloading. In *Proceedings of the 3rd IEEE Workshop on Internet Applications* (pp. 79–89).
15. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., & Zhang, X. (2007). A performance study of BitTorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, *25*(1), 155–169.
16. Koo, S. G. M., Kannan, K., & Lee, C. S. G. (2006). On neighbor-selection strategy in hybrid peer-to-peer networks. *Future Generation Computer Systems*, *22*, 732–741.
17. Koo, S. G. M., Rosenberg, C., & Xu, D. (2003). Analysis of parallel downloading for large file distribution. In *Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems* (pp. 128–135).
18. Kumar, R., & Ross, K. W. (2006). Peer-assisted file distribution: The minimum distribution time. In *Proceedings of the IEEE Workshop on Hot Topics in Web Systems and Technologies*.
19. Li, K. (2012). Probing high-capacity peers to reduce download times in P2P file sharing systems with stochastic service capacities. *International Journal of Foundations of Computer Science*, *23*(6), 1341–1369.
20. Li, K. (2013). Parallel file download in peer-to-peer networks with random service capacities. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium Workshops* (15th Workshop on Advances in Parallel and Distributed Computational Models), pp. 677–686, Boston, MA, May 20–24.
21. Li, K. (2014). On the expected file download time of the random time-based switching algorithm in P2P networks. *Peer-to-Peer Networking and Applications*, *7*(2), 147–158.
22. Li, K. (2015). Analysis of file download time in peer-to-peer networks with stochastic and time-varying service capacities. *Future Generation Computer Systems*, *42*, 36–43.
23. Lingjun, M., & Lui, K.-S. (2008). Scheduling in P2P file distribution—on reducing the average distribution time. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference* (pp. 521–522).
24. Lingjun, M., Xiaolei, W., & Lui, K.-S. (2008). A novel peer grouping scheme for P2P file distribution networks. In *Proceedings of the IEEE International Conference on Communications* (pp. 5598–5602).
25. Liu, Y., Gong, W., & Shenoy, P. (2001). On the impact of concurrent downloads. In *Proceedings of the 33nd Winter Simulation Conference* (pp. 1300–1305).
26. Manini, D., & Gribaudo, M. (2006). Modelling search, availability, and parallel download in P2P file sharing applications with fluid model. In *Proceedings of 14th International Conference on Advanced Computing and Communications* (pp. 449–454).
27. Rodriguez, P., & Biersack, E. W. (2002). Dynamic parallel access to replicated content in the internet. *IEEE/ACM Transactions on Networking*, *10*(4), 455–465.
28. Stutzbach, D., & Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*.
29. Teo, M., Carbunaru, C., Leong, B., Nataraj, Y., Vu, H. M. L., Tan, R., & Teo, Y. M. (2008). Achieving high-bandwidth peer-to-peer file distribution. In *Proceedings of the 4th ACM International Conference on Emerging Networking Experiments and Technologies*.
30. Tewari, S., & Kleinrock, L. (2005). On fairness, optimal download performance and proportional replication in peer-to-peer networks. In *Proceedings of the 4th International IFIP-TC6 Networking Conference (LNCS 3462)* (pp. 709–717).
31. Zheng, X., Cho, C., & Xia, Y. (2008). Optimal peer-to-peer technique for massive content distribution. In *Proceedings of the 27th IEEE Conference on Computer Communications* (pp. 151–155).
32. Zwillinger, D. (Ed.). (1996). *Standard mathematical tables and formulae* (30th ed.). Boca Raton, FL: CRC Press.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 430 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, *Journal of Parallel and Distributed Computing*. He is an IEEE Fellow.