INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Simulation Analysis of Download and Recovery Processes in P2P Storage Systems

Abdulhalim Dandoush — Sara Alouf — Philippe Nain

N° 6858

Février 2009

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6858--FR+ENG

Simulation Analysis of Download and Recovery Processes in P2P Storage Systems

Abdulhalim Dandoush , Sara Alouf , Philippe Nain

Thème COM — Systèmes communicants
Projet MAESTRO

Rapport de recherche n° 6858 — Février 2009 — 19 pages

Abstract: Peer-to-peer storage systems rely on data fragmentation and distributed storage. Unreachable fragments are continuously recovered, requiring multiple fragments of data (constituting a "block") to be downloaded in parallel. Recent modeling efforts have assumed the recovery process to follow an exponential distribution, an assumption made mainly in the absence of studies characterizing the "real" distribution of the recovery process. This report aims at filling this gap through an empirical study. To that end, we implement the distributed storage protocol in the NS-2 network simulator and run a total of six experiments covering a large variety of scenarios. We show that the fragment download time follows approximately an exponential distribution. We also show that the block download time and the recovery time essentially follow a hypo-exponential distribution with many distinct phases (maximum of as many exponentials). We use expectation maximization and least square estimation algorithms to fit the empirical distributions. We also provide a good approximation of the number of phases of the hypo-exponential distribution that applies in all scenarios considered. Last, we test the goodness of our fits using statistical (Kolmogorov-Smirnov test) and graphical methods.

Key-words: Peer-to-Peer, storage systems, performance evaluation, download time, Recovery process, hypo-exponential distribution, data fitting

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Analyse par simulation des temps de téléchargement et de réparation des données dans les systèmes de stockage pair-à-pair

Résumé : Les systèmes pair-à-pair de stockage de données reposent sur la fragmentation des données et le stockage décentralisé. Les fragments devenus inaccessibles sont constamment récupérés, ce qui nécessite le téléchargement en parallèle de plusieurs fragments (constituant "un bloc"). Des études récentes de nature théorique supposent que le processus de récupération suit une distribution exponentielle, une hypothèse faite surtout faute d'études caractérisant la distribution "réelle" du processus de récupération. Ce rapport vise à combler cette lacune par une étude empirique. Tout d'abord, nous implémentons le protocole de stockage distribué dans le simulateur de réseau NS-2. Nous effectuons une série de simulations couvrant une grande variété de scénarios. Nous montrons par la suite que le temps de téléchargement d'un fragment suit plutôt une distribution exponentielle. Nous montrons aussi que le temps de téléchargement d'un bloc et le temps de réparation des données suivent plutôt une distribution hypo-exponentielle ayant plusieurs phases distinctes. Nous utilisons la méthode des moindres carrés et l'algorithme d'espérance-maximisation pour ajuster les distributions empiriques avec des distributions statistiques. Nous trouvons aussi une bonne approximation du nombre de phases de la distribution hypo-exponentielle. Pour finir, nous effectuons le test d'hypothèse de Kolmogorov-Smirnov sur les distributions statistiques identifiées pour corroborer nos conclusions.

Mots-clés : Systèmes pair-à-pair, systèmes de stockage, évaluation de performance, temps de téléchargement, réparation des données, distribution hypo-exponentielle

Contents

1 Introduction

The peer-to-peer (P2P) model has proved to be an alternative to the Client/Server model and a promising paradigm for Grid computing, file sharing, voice over IP, backup and storage applications. A major advantage of P2P systems is that peers can build a virtual overlay network on top of existing architecture and topology. Each peer receives/provides a service from/to other peers through the overlay network; examples of such a service are sharing the capacity of its central processing unit, sharing its bandwidth capacity, sharing its free storage space, and sharing local information about neighbors to help each other locating resources.

P2P storage systems (P2PSS) have emerged as a cheap, scalable and self-repairing solution. Such distributed systems rely on data fragmentation and distributed storage. Files are partitioned into fixed-size blocks that are themselves partitioned into fragments. Fragments are usually stored on different peers. Given this configuration, a user wishing to retrieve a given data would need to perform multiple downloads, generally in parallel for an enhanced service. To mitigate churn of peers, redundant fragments are continuously injected in the system, thus maintaining data redundancy above a minimum desired level. When the amount of unreachable fragments attains a predefined threshold, a *recovery process* is initiated.

In this report, we consider systems relying on erasure codes to generate the redundant fragments. If s denotes the initial number of fragments and r denotes the amount of additional redundant fragments, then any s out of the $s + r$ fragments can be used to generate a new redundant fragment (e.g.

[20]). Observe that this notation covers the case of replication-based systems, with $s = 1$ and r denoting the number of replicas.

The recovery process includes the download of a full block of s fragments. P2PSS may rely on a central authority that initiates the recovery process when necessary. This central authority could reconstruct all missing fragments of a given block of data and remotely store them on as many new peers.[1] Alternatively, secure agents running on new peers could reconstruct by themselves missing fragments to be stored on the peers disks. A more detailed description of P2PSS, their recovery schemes and their policies is presented in Section 2.

1.1 Motivation

There have been recent modeling efforts focusing on the performance analysis of P2PSS in terms of data durability and data availability. In [19], Ramabhadran and Pasquale analyze systems that use *full replication* for data reliability. They develop a Markov chain analysis, then derive an expression for the lifetime of the replicated state and study the impact of bandwidth and storage limits on the system. This study relies on the assumption that the recovery process follows an exponential distribution. Observe that in replication-based systems, the recovery process lasts mainly for the download of one fragment of data. In other words, the authors of [19] are implicitly assuming that the fragment download time is exponentially distributed.

In our previous work [2], we developed a more general model than that in [19], which applies to both replicated and erasure-coded P2PSS. Also, unlike [19], our model accounts for transient disconnections of peers, namely, the churn in the system. We also assumed the recovery process to be exponentially distributed. However, this assumption differs substantially between replicated and erasure-coded P2PSS, as in the latter systems the recovery process is much more complex than in the former systems. Furthermore, the recovery process differs from centralized to distributed implementation.

More recently (see [6]), we have fine-tuned the model of [2] by assuming that both fragment download and upload times are exponentially distributed with parameters α and β respectively. This assumption is in line with that made in [19].

In all above-mentioned studies, findings and conclusions rely on the assumption that either the fragment download time or the recovery process or both are exponentially distributed. However, none of these assumptions are backed up by experimental data. To the best of our knowledge, there has been no empirical study characterizing these processes in real P2PSS.

This work aims at filling this gap through a simulation analysis. We believe it is essential to characterize the distribution of download and recovery processes in P2PSS. Evaluating these distributions is crucial to validate (or invalidate) the results presented in the above works and to better understand the availability and durability of data in these systems.

We will show through intensive simulations of many realistic scenarios that the fragment download time follows closely an exponential distribution. Given that in erasure-coded systems, the block download time consists of downloading several fragments in parallel, it follows that the recovery pro-

[1] By "new" peers, we refer to peers that do not already store fragments of the same block. We assume the system enforces the rule that a peer can store at most one fragment of any given block.

cess should follow approximately a hypo-exponential distribution of several phases. (This is nothing but the sum of several random variables exponentially distributed having each its own rate [12].) We will show that this is indeed the case in the simulated data. We find that the exponential assumption made on the block download time and on the recovery process is not met in most cases that we considered. Therefore, our results support the assumptions made in [19] and [6].

1.2 Why do we use simulations?

To collect traces of fragment download/upload times, of block download times and of recovery times, one can choose to perform simulations or experimentations either on testbeds or on real networks. We would like to consider situations where peers are either homogeneous or heterogeneous, different underlying network topologies, and different propagation delays in the network. Also, we would like to consider systems with a large number of peers. To achieve all this with experiments over real networks is very difficult. Setting up experiments over a dedicated network like Planet-Lab [18] would require a long time, and there will be limitations on changing the topology and the peers characteristics.

We find it most attractive to implement the distributed storage protocol in a well-known network simulator and to simulate different scenarios. We choose NS-2 as network simulator because it is an open source discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP and routing and it is well known and well validated.

1.3 Contributions

Our contributions are as follows:

- Implementation of the download and the recovery processes in NS-2.
- Evaluation of the fragment/block download time and the recovery process, under a variety of conditions: different network topologies, heterogeneity of peers, different propagation delay, centralized vs. distributed recovery process.
- Data fitting: the distribution of the block download time and recovery time are fitted using the Expectation Maximization (EM) algorithm and the distribution of the fragment download time is fitted using the Maximum Likelihood Estimation (MLE) and Least Square Estimation (LSE).
- Statistical goodness-of-fit test, namely, the Kolmogorov-Smirnov test [15].

The rest of this report is organized as follows. Section 2 overviews the storage protocol that we consider. Section 3 describes the simulation architecture, the methodology and the setup of the simulations. In Section 4, some of our experimental results are discussed. Section 5 briefly reviews related work. Last, Section 6 concludes the report.

2 System Description

We will describe in this section the storage protocol that we want to simulate:

- Files are partitioned into fixed-size blocks (the block size is S_B) that are themselves partitioned into s fragments (the fragment size is S_F).
- Each block is stored as a total of $s+r$ fragments, r of them are redundant and generated using erasure codes.
- Fixing block and fragment sizes helps to fix the value of the parameters s and r in the system for all stored blocks. These $s + r$ fragments are stored over $s + r$ different peers.
- Mainly for privacy issues, a peer can store at most one fragment of any block of data.
- The system has perfect knowledge of the location of fragments at any given time, e.g. by using a Distributed Hash Table (DHT) or a central authority. Only the latest known location of each fragment is tracked, whether it is a connected or disconnected peer.
- To overcome churn and maintain data reliability and availability, unreachable fragments are continuously recovered.
- The number of connected peers at any time is typically much larger than the number of fragments associated with a block of data, i.e., $s + r$. Therefore, there are always at least $s + r$ *new* connected peers which are ready to receive and store fragments of a block of data.
- Once an unreachable fragment is recovered, any other copy of it that "reappears" in the system due to a peer reconnection is simply ignored, as only one location of the fragment (the newest one) is recorded in the system. Similarly, if a fragment is unreachable, the system knows of only one disconnected peer that stores the unreachable fragment.

Two implementations of the recovery process are considered. This process is triggered for each block whose number of unreachable fragments reaches a threshold k.

In the centralized implementation, a central authority will: (1) download *in parallel* s fragments from the peers which are connected, (2) reconstruct at once all unreachable fragments (by now considered as missing), and (3) upload them all *in parallel* onto as many new peers for storage. In fact, Step 2 executes in a negligible time compared to the execution time of Steps 1 and 3. Step 1 (resp. Step 3) execution completes when the last fragment completes being downloaded (resp. uploaded).

In the distributed implementation, a secure agent on one new peer is notified of the identity of *one* out of the k unreachable fragments for it to reconstruct it. Upon notification, the secure agent (1) downloads s fragments from the peers which are connected to the storage system, (2) reconstructs the specified fragment and stores it on the peer's disk; (3) the secure agent then discards the s downloaded fragments so as to meet the privacy constraint that only one fragment of a block of data is held by a peer. This operation iterates until less than k fragments are sensed unreachable and stops if the number of missing fragments reaches $k - 1$. The recovery of one fragment lasts mainly for the execution time of Step 1; the recovery is completed then as soon as the last fragment (out of s) completes being downloaded.

When $k = 1$, the recovery process is said to be *eager*; when $k \in \{2, \dots, r\}$, the recovery process is said to be *lazy*.

3 Simulation Architecture and Setup

3.1 Architecture and Assumptions Overview

We implemented an application and a wrapper layers in NS-2 (version 2.33) following the architecture depicted in Figure 1. The application layer represents the P2PSS application. As for the wrapper layer, it is an intermediate layer that passes the data between a transport agent object in NS-2 and the P2PSS application.

We did minor changes to the following NS-2 files: node.cc, node.h, agent.cc, agent.h, tcp-full.cc, and tcp-full.h. We use FullTcp since it supports bidirectional data transfers. We follow the same methodology as the Web cash application presented in the NS Manual (cf. [9, Chap. 40]) and use some of the technical ideas presented in [8].

Implementing a new protocol at the application level of NS-2 is very well documented in the NS Manual. We will therefore skip the description of technical details of the implementation, and refer the interested reader to [9, pp. 344–360].

We consider two different storage applications, a backup-like application and an e-library-like application ("e" stands for "electronic") . In the first, a file stored in the system can be requested for retrieval only by the peer that has produced the file. In the second, any file can be downloaded by any peer in the system. In both applications, the storage protocol follows the description of Section

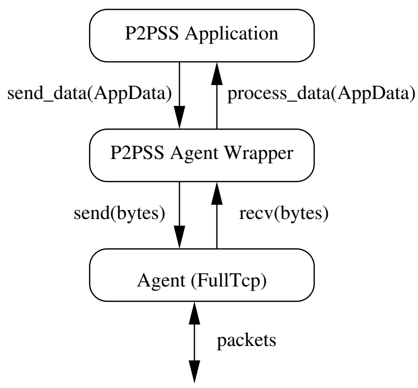


Figure 1: Simulator architecture.

2. In particular, the $s + r$ peers associated with a given block are chosen uniformly among the peers in the system.

Two types or requests are issued in the system. The first type is issued by the users of the system: a user issues a request to retrieve one of its files in the backup-like application, or a public document in the e-library-like application. The second type consists in management requests. These are issued by the central authority (in the centralized implementation of the recovery process) or by a peer (in the distributed implementation) as soon as the threshold k is reached for any stored block of data.

File download requests are translated into (i) a request to the directory service to obtain, for each block of the desired file, a list of at least s peers that store fragments of the block, (ii) opening TCP connections with each peer in the said list to download one fragment. All download requests issued by a given peer form a Poisson process.

Recovery requests are issued only in the scenarios where there is churn in the network. A recovery request concerning a given block translates into (i) a request to the directory service to obtain a list of at least s peers that store fragments of said block, (ii) opening TCP connections with each peer in the said list to download one fragment. Once all s fragments have been downloaded, the process proceeds with Steps 2 and 3, according to the implementation, as explained in Section 2.

All peers in the simulator have the architecture reported in Figure 1. Peers share their available upload bandwidths and their free storage volumes.

3.2 Network Topology

Having a representative view of enterprise networks or the Internet topology is very important for a simulator to predict the behavior of a network protocol or application if it were to be deployed. In fact, the simulated topology often influences the outcome of the simulations. Realistic topologies are thus needed to produce realistic simulation results. Most of existing simulations studies have used representations of a real topology (e.g. the Arpanet), simple models (e.g. a star topology), or random flat graphs (i.e. non-hierarchical) that are generated by Waxman's edge-probability function [22].

However, random models offer very little control over the structure of the resulting topologies. In particular, they do not capture the hierarchy that is present in the Internet. Recently, tools such as (BRITE [16] and GT-ITM [4]) have been designed to generate more complex random graphs, that are hierarchical, to better approximate the Internet's hierarchical structure.

To produce realistic topologies for our simulations, we use the tool GT-ITM [4] to generate a total of six random graphs. Three levels of hierarchy are used corresponding to transit domains, stub domains, and local area networks (LANs) attached to stub domains. Each graph has one transient domain of four nodes; each of the nodes is connected to two or three other transit nodes. Each transit node is connected on average to two stub nodes, and each stub node is in turn connected on average to four routers. Behind every router there is a certain number of fully-connected peers constituting a LAN. The first of these six graphs is depicted in Figure 2, where we have used the notation TN for "transit node" and SN for "stub node". The total number of peers in each graph is in the set $\{480, 640, 800, 960\}$.

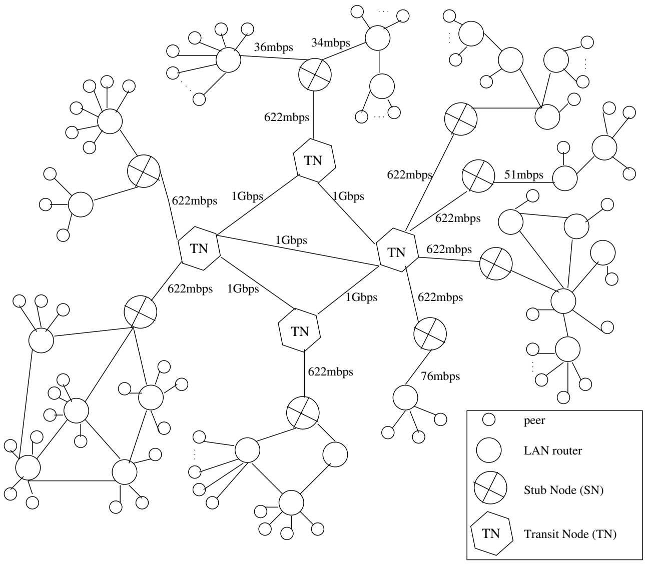


Figure 2: Three-level hierarchical random graph of Experiment 1.

3.3 Experiments Setup

We ran a total of seven experiments. Experiments 1–5 used the random graphs generated with the GT-ITM tool as detailed before, whereas a simple star topology is used in Experiment 6. Regarding the intra- and inter-domain capacities, we rely on the information provided by RENATER [21] and GÉANT [10] web sites. In those networks, the links are well-provisioned. To have a more complete study, we will consider, in Experiments 4 and 5, links with smaller capacities, as can be seen in rows 4–6 of Table 1. Propagation delays over TN-SN edges vary from edge to edge as can be seen in row 7 of Table 1.

Let C_u and C_d denote respectively the upload and download capacity of a peer. To set these values, we rely mainly on the findings of [11] and [14]. The experimental study of file sharing systems and of the Skype P2P voice over IP system [11] found that more than 90% of users have upload capacity C_u between 30Kbps and 384Kbps. However, the measurement study [14] done on BitTorrent clients in 2007 reports that 70% of peers have an upload capacity C_u between 350Kbps and 1Mbps and even 10% of peers have an upload capacity between 10Mbps and 110 Mbps. The capacities that we have selected in the simulations vary between the values of the ISDN and ADSL technologies; they can be found in rows 8–9 of Table 1. Observe that across all experiments peers are heterogeneous. We will attribute the propagation delays over routers-peers edges randomly between 1ms and 25ms as can be seen in row 10 of Table 1.

Table 1: Experiments setup

Experiment number	1	2	3	4	5	6
Topology	random	random	random	random	random	star
Number of peers	960	480	960	640	800	480
TN-TN capa. (Gbps)	1	1	1	1	1	—
TN-SN capa. (Mbps)	622	622	622	10–34	10–34	—
SN-R capa. (Mbps)	34–155	34–155	34–155	4–10	4–10	—
TN-SN delays (ms)	5–25	5–75	5–50	5–25	5–25	—
C_u of peers (Kbps)	150–1000	128–1000	128–1000	256–700	256–1000	256–700
C_d of peers (Kbps)	$8 \times C_u$	$8 \times C_u$	$8 \times C_u$	$10 \times C_u$	$4 \times C_u$	2048
R-peers delays (ms)	1–20	1–20	1–20	1–10	1–25	1–25
Background traffic	yes	yes	no	yes	yes	no
Application type	e-library	backup	e-library	backup	e-library	e-library
Peers churn	no	no	no	yes	yes	yes
Recovery process	—	—	—	dist.	dist.	cent.
r	—	—	—	s	s	$s/2$
$1/\lambda$ (min.)	80	80	144e3	160	13	16
S_B (MB)	8	8	8	4	8	8
S_F (KB)	1024	1024	1024	512	1024	1024
s	8	8	8	8	8	8

In Experiments 1, 2, 4 and 5, there exists a background traffic between three pairs of routers across the common backbone. This traffic consists in random exponential and CBR traffic over UDP protocol and FTP traffic over TCP.

In each of the experiments, the amount of data transferred between routers and peers in the system during the observed time (that is from 4e+5 up to 5e+6 seconds) are, on average, 4.5–9 GB of P2P application traffic, and when applicable 150–350 MB of FTP, 200–400 MB of CBR, and 250–500 MB of the exponential traffic.

Experiments 2 and 4 simulate a backup-like application whereas the other five experiments simulate an e-library-like application. Churn is considered only in Experiments 4–6. As a consequence, redundancy is added and maintained only in these experiments. The storage overhead r/s is either 1 or 0.5. We consider the distributed implementation of the recovery process in Experiments 4 and 5, and the centralized implementation of the same in Experiment 6; the eager policy ($k = 1$) is considered in all three experiments. In other words, once a peer disconnects from the system, all fragments that are stored on it must be recovered.

Churn is implemented as follows. We assume that each peer alternates between a connected state, that lasts for a duration called “on-time”, and a disconnected state, that lasts for a duration called “off-time”. We assume in the simulations that the successive on-times (respectively off-times) of a peer are independent and identically distributed random variables with a common exponential distribution function with parameter $\mu_1 > 0$ (respectively $\mu_2 > 0$). This assumption is in agreement with the analysis in [19]. We consider $1/\mu_1 = 3$ hours and $1/\mu_2 = 1$ hours.

Download requests are generated at each peer according to a Poisson process. This assumption is met in real networks as found in [11]. We assume all peers have the same request generation rate, denoted λ. We vary the value of λ across the experiments as reported in row 16 of Table 1.

The last setting concerns the files that are stored in the P2PSS. Fragment sizes S_F (resp. block sizes S_B) in P2P systems are typically between 64KB and 4MB each (resp. between 4MB and 9MB each). We will consider in most of our experiments $S_F = 1$MB and $S_B = 8$MB, except in Experiment 4 where $S_F = 512$KB and $S_B = 4$MB. Therefore $s = 8$ is all experiments. As for the file size, we assume for now that it is equal to the block size. Therefore, the file download size is actually the block download size. We leave the case of more general file sizes to a future study. Observe that the recovery process is related to the block download time and not to the file download time.

Table 1 summarizes the key settings of the experiments.

4 Experimental Results

In this section, we present the results of our simulations and the inference that we can draw from them. For each experiment, we collect the fragment download time, the block download time and the recovery time when applicable. In Experiments 4 and 5 (distributed recovery), the two latter durations are collected to the same dataset as there is no essential difference between them. Having collected these samples, we compute the sample average and use MLE, LSE and EM algorithms to fit the empirical distributions. Concerning the fragment download time, we perform the Kolmogorov-Smirnov test [15] on the fitted distribution. In the following, we will present selected results from Experiments 1, 5 and 6. The results of the other experiments are briefly reported in Tables 2–3.

4.1 Experiment 1

We have collected 76331 samples of the fragment download time (cf. column 2 of Table 2). The empirical cumulative distribution function (CDF) is depicted in Figure 3(a). We can see that it is remarkably close to the exponential distribution. Two exponential distributions are plotted in Figure 3(a), each having a different parameter, derived from a different fitting technique. The two techniques that we used are MLE and LSE. The parameter returned by MLE is nothing but the inverse of the sample average and is denoted α; see row 2 of Table 2.

Beyond the graphical match between the empirical distribution and the exponential distribution, we did a hypothesis test. Let $\mathbf{X}$ be a vector storing the collected fragment download times. The Kolmogorov-Smirnov test compares the vector $\mathbf{X}$ with a CDF function, denoted cdf (in the present case, it is the exponential distribution), to determine if the sample $\mathbf{X}$ could have the hypothesized continuous distribution cdf. The null hypothesis is that $\mathbf{X}$ has the distribution defined in cdf, the alternative one being that $\mathbf{X}$ does not have that distribution. We reject the null hypothesis if the test is significant at the $l\%$ level. In Experiment 1, the null hypothesis with $\alpha = 1/40.35$ is not rejected for $l = 7\%$.

Table 2: Summary of experiments results

Experiment number	1	2	3	4	5	6
Av. frag. down. time $= 1/\alpha$	40.35	44.89	30.66	34.7367	108.86	40.722
Samples number	76331	12617	4851	9737	80301	4669
t_m (sec.)	8.77	8.631	8.71	6.84	8.743	16.4
$1/\hat{\alpha}$ (sec.)	39.351	39.622	27.3392	32.106	103.635	32.05
$1/\beta$, $1/\hat{\beta}$ (sec.)	—	—	—	—	—	6.22, 5.11
Av. recov. or block down. time	102.75	105.254	82.88	92.4762	278.71	89.848
Samples number	9197	1516	602	589	10025	561

Table 3: Block download time or recovery process: Validation of the approximations introduced in Eqs. (1)–(4)

Experiment number	1	2	3	4	5	6
Sample average	102.75	105.25	82.88	92.48	278.71	89.85
Inferred average from Eqs. (1), (2)	109.66	122.00	83.33	94.40	295.86	116.89
Relative error (%)	6.7	15.9	0.5	2.1	6.2	30.1
Inferred average from Eqs. (3), (4)	106.95	116.32	83.01	94.10	290.41	92.21
Relative error (%)	4.1	10.5	0.2	1.8	4.2	2.6

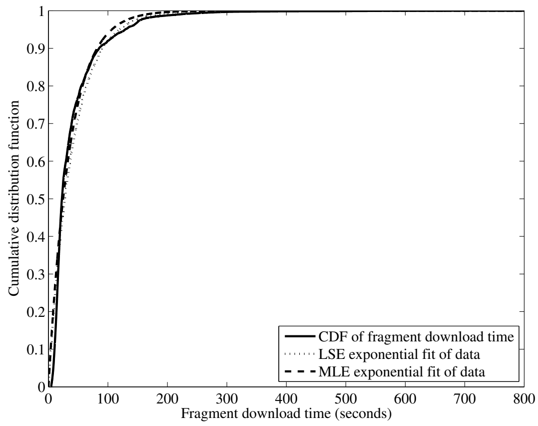
Regarding the block download times, we have collected 9197 samples. The sample average is given in row 7 of Table 2). The empirical CDF is plotted in Figure 3(b). We followed the same methodology and computed the closest exponential distribution using MLE. However, the match between the two distribution appears to be poor, and actually, the alternative hypothesis is not rejected in this case.

To find a distribution that will more likely fit the empirical data, we make the following analysis. To get a block of data, s fragments, stored on s different peers, have to be downloaded. This is more efficiently done in parallel and this is how we implemented it in the simulator. We have seen that the download of a single fragment is well-modeled by an exponential random variable with parameter α. Therefore, downloading s fragments in parallel is distributed like the maximum of s exponential random variables. Assuming these downloads to be mutually independent—assumption not necessarily met in the simulations—the maximum is then the sum of s independent exponential random variables with parameters $s\alpha, (s-1)\alpha, \ldots, \alpha$, due to the memoryless property (see also [12]). This distribution is called the hypo-exponential distribution and its expectation is

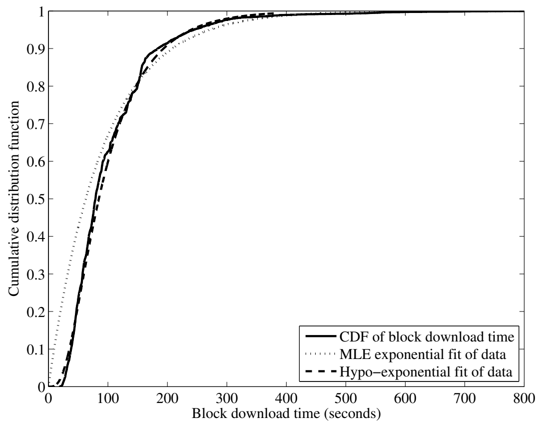
$$E[T] = 1/\alpha \sum_{i=1}^{s} 1/i \tag{1}$$

where T denotes the block download time (or equivalently the distributed recovery duration).

In Experiment 1, $E[T] = 109.66$ seconds, while the sample average is equal to 102.75; cf. column 2 of Table 3. The relative error is 6.7%. The hypo-exponential distribution with s phases



(a) Exponential fit of the fragment download time distribution



(b) Fitting of the block download time distribution

Figure 3: Experiment 1: Fragment and block download times.

and parameters $s\alpha, (s-1)\alpha, \ldots, \alpha$ is plotted in Figure 3(b). This distribution has a very good visual match with the empirical CDF of the block download time.

As a next step, we apply an EM algorithm [7] to find the best hypo-exponential distribution with s phases that fits the empirical data. In particular, we use EMpht [17], which is a program for fitting phase-type distributions to collected data. We do not plot the outcome of this program in Figure 3(b) as it mainly overlaps with the hypo-exponential distribution with s phases and parameters $s\alpha, (s-1)\alpha, \ldots, \alpha$ that is already plotted there. After performing the Kolmogorov-Smirnov test, we find that the null hypothesis is not rejected for $l = 7\%$ (same significance level as for the fragment download times).

We conclude the analysis of the first experiment's results with three important points:

- The exponential assumption on the block download time is not met in realistic simulations.
- The fragment download time could be modeled by an exponential distribution with parameter α equal to the inverse of its average.
- As a consequence, the block download time could be modeled by a hypo-exponential distribution with s phases and parameters $s\alpha, (s-1)\alpha, \ldots, \alpha$.

4.2 Experiment 5

In this experiment, peers are not always connected. Each time a peer disconnects from the network, all the fragments that were stored on his disk will have to be recovered. The recovery process is implemented in a distributed way.

The empirical CDF of the fragment download time and that of the block download time or the recovery time are reported in Figure 4. Following the same methodology as that used to analyze the results of Experiment 1, we reach the same conclusions. The relevant parameters are reported in column 6 of Tables 2 and 3. However, the null hypothesis for the block download time or the recovery process is not always rejected. This is the case of Experiment 6, as seen next.

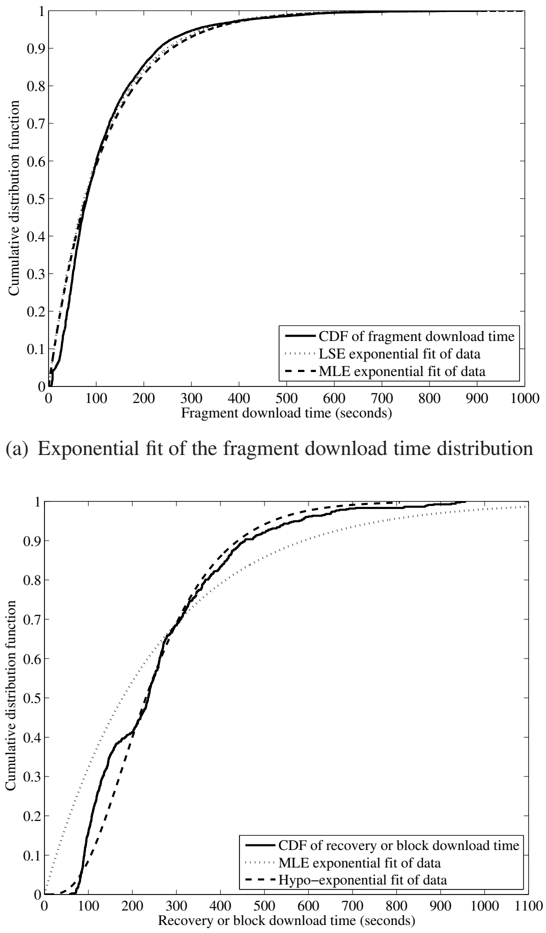
4.3 Experiment 6

Experiment 6 is the only one that uses a centralized recovery process. Also, it is the only one using a simple star topology. In this experiment, the alternative hypothesis on the recovery process distribution is not rejected.

There is a simple reason for that. We actually know that the download of a single fragment cannot be infinitely small, as suggested by the exponential distribution. Let t_m be the duration of the fastest fragment download among all s downloads. All other (slower) downloads are necessarily bounded by t_m. The effect of this minimum value can be neglected as long as t_m is negligible with respect to the average fragment download time. Otherwise, we need to consider that the fragment download/upload time is composed of two components: (i) a (constant) minimum delay t_m and (ii) a random variable distributed exponentially with parameter $\hat{\alpha}$ (resp. $\hat{\beta}$). This random variable models the collected data, shifted left by the value of t_m. The minimum delay can be approximated as $RTT + (S_F + \text{ Headers})/\max\{C_u\}$, where RTT stands for round-trip time.

The value of t_m is clearly visible in Figure 5(a). We plot in this figure the empirical CDF of the fragment download time, the MLE exponential fit to the collected data and the MLE exponential fit to the shifted data. The null hypothesis is not rejected in both cases.

This is not the case of the recovery process, whose empirical CDF is plotted in Figure 5(b). Repeating the same analysis than in Section 4.1, and assuming that the fragment upload time follows an exponential distribution with parameter β, then the centralized recovery process, denoted T_c,



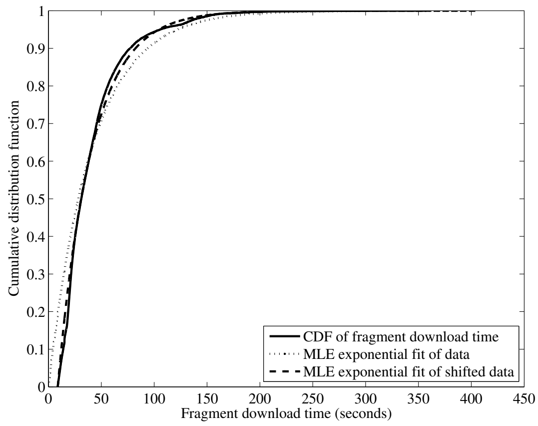
(a) Exponential fit of the fragment download time distribution

(b) Fitting of the recovery or block download time distribution

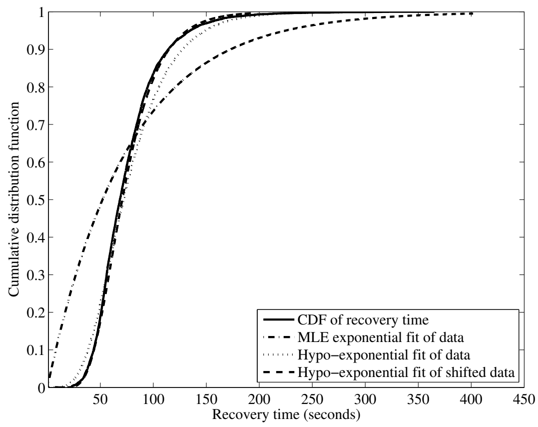
Figure 4: Experiment 5: Download and distributed recovery processes.

would be modeled by a hypo-exponential distribution with $s + k$ phases ($k = 1$ in Experiment 6) having expectation

$$E[T_c] = 1/\alpha \sum_{i=1}^{s} 1/i + 1/\beta \sum_{j=1}^{k} 1/j \,. \tag{2}$$



(a) Exponential fit of the fragment download time distribution ignoring the minimum value



(b) Fitting of the recovery time distribution

Figure 5: Experiment 6: Fragment and recovery time, centralized recovery.

Considering this distribution, we find that the null hypothesis of the Kolmogorov-Smirnov test for the collected data with parameters $1/\alpha = 40.72$ and $1/\beta = 6.22$ is rejected[2] for $l = 6\%$, while it is not rejected for the shifted data with parameters $1/\hat{\alpha} = 32.05$ and $1/\hat{\beta} = 5.11$.

[2]Even though it is rejected, this distribution is still much closer to the empirical data than the exponential distribution.

Equations (1) and (2) should then be replaced with

$$E[T] = t_m + 1/\hat{\alpha} \sum_{i=1}^{s} 1/i\,, \tag{3}$$

$$E[T_c] = t_m + 1/\hat{\alpha} \sum_{i=1}^{s} 1/i + 1/\hat{\beta} \sum_{j=1}^{k} 1/j\,. \tag{4}$$

The averages inferred from Eqs. (1)–(4) are listed in rows 3 and 5 of Table 3, and their relative errors with respect to the sample average are listed in rows 4 and 6 of the same table. Observe that the inferred average improves across all experiments when considering shifted data. The best improvement seen is that in Experiment 6. By considering that the *shifted* recovery time is hypo-exponentially distributed with $s+1$ phases and parameters $s\hat{\alpha}, (s-1)\hat{\alpha}, \ldots, \hat{\alpha}, \hat{\beta}$, the relative error on the inferred average drops from 30.1% to 2.6%.

The conclusion of this discussion is that the exponential assumption on fragments download/upload time is met in most cases. The same assumption does not hold on the block download time. The recovery time and the block download time are well approximated by a hypo-exponential distribution in most cases.

5 Related work

Although the literature on modeling and simulating P2P systems and parallel downloading is abundant, the recovery process in P2PSS is a subject that has not been analyzed.

In [3], the authors propose a multiple-access protocol to minimize the download time of a document from multiple mirror sites in parallel using Tornado erasure codes based on the idea of digital fountain. A document of size S_B is encoded on each mirror server with redundant information. The encoded document consists of $n = s + r$ different fragments of size S_F where $nS_F > sS_F > S_B$. To minimize the number of duplicated packets received at the requester, each mirror encodes the document with Tornado codes and generates all the n fragments, then it permutes the order of packets before sending, and finally starts to deliver the packets continuously to the requester of the document. The receiver can then reconstruct the document S_B after collecting s distinct packets of size S_F from the mirrors. In [5], the authors have focused on the average download time of each user in a P2P network while considering the heterogeneity of service capacities of peers. They point out that the common approach of analyzing the average download time based on average service capacity is fundamentally flawed.

K. Eger et al. implement in [8] a BitTorrent file sharing protocol in NS-2 and compare packet-level simulation results with flow-level for the download time of one file among an active peer-set. They show that the propagation delay can significantly influence the download performance of BitTorrent. V. Aggarwal et al. implement in [1] a Gnutella file sharing protocol in SSFNet simulator and compare also the packet-level and the flow-level simulation results. The user behavior model is reflected in their simulation framework. Q. He et al. implement in [13] a framework of P2P file sharing, in particular Gnutella, for NS, and show how Gnutella system performance can be impacted by the network characteristics.

6 Conclusion

This report performs a simulation analysis of download and recovery processes in P2PSS. Implementing a storage protocol in NS-2, we set up seven simulations which enables us to collect fragment/block download times and recoveries times under a variety of conditions. We show that the exponential assumption on the block download time does not hold. The same assumption on fragments download/upload time is met in most cases implying that the both the block download time and the recovery process could be modeled by a hypo-exponential phases with a pre-determined number of phases.

References

[1] V. Aggarwal, O. Akonjang, A. Feldmann, R. Tashev, and S. Mohrs. Reflecting p2p user behaviour models in a simulation environment. In *Proc. of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 16–523, Washington, DC, USA, 2008.

[2] S. Alouf, A. Dandoush, and P. Nain. Performance analysis of peer-to-peer storage systems. In *Proc. of 20th ITC*, volume 4516 of *Lecture Notes in Computer Science*, pages 642–653, Ottawa, Canada, June 2007.

[3] J. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *Proc. of IEEE Infocom '99*, pages 21–25, New York, USA, March 1999.

[4] K. Calvert, M. Doar, and E. W. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, June 1997.

[5] Yuh-Ming Chiu and Do Young Eun. Minimizing file download time in stochastic peer-to-peer networks. *IEEE/ACM Trans. Netw.*, 16(2):253–266, 2008.

[6] A. Dandoush, S. Alouf, and P. Nain. Performance analysis of centralized versus distributed recovery schemes in P2P storage systems. In *Proc. of IFIP/TC6 NETWORKING 2009*, Aachen, Germany, May 11–15, 2009. to appear.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statist. Soc.*, 39(1):1–37, 1977.

[8] K. Eger, T. Hobfeld, A. Binzenhofer, and G. Kunzmann. Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations. In *Proc. of UPGRADE-CN'07*, Monterey, California, USA, June 2007.

[9] K. Fall and K. Varadhan. The NS manual, the VINT project, UC Berkeley, LBL, USC/ISI, and Xerox PARC. `http://www.isi.edu/nsnam/ns/ns-documentation.html`, November 2008.

[10] GÉANT: a pan-European backbone which connects Europe's national research and education networks. `http://www.geant.net/server/show/nav.159`.

[11] A. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer VoIP system. In *Proc. of 5th IPTPS*, Santa Barbara, California, February 2006.

[12] P. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation Journal*, 64(7-8):664–689, August 2007.

[13] Qi. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto. Mapping peer behavior to packet-level details: A framework for packet-level simulation of peer-to-peer systems. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, Orlando, Florida, USA, October 2003.

[14] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Leveraging Bittorrent for end host measurements. In *Proc. 8th Passive and Active Measurement Conference*, Louvain-la-neuve, Belgium, April 2007.

[15] F. J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *J. Am. Statist. Assoc.*, 46(253):68–78, 1951.

[16] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Boston University representative Internet topology generator. `http://www.cs.bu.edu/brite/`.

[17] M. Olsson. The EMpht-programme. Technical report, Department of Mathematics, Chalmers University of Technology, 1998.

[18] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`, 2007.

[19] S. Ramabhadran and J. Pasquale. Analysis of long-running replicated systems. In *Proc. of IEEE Infocom '06*, Barcelona, Spain, April 2006.

[20] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8(2):300–304, June 1960.

[21] Renater: Le Réseau National de télécommunications pour la Technologie, l'Enseignement et la Recherche. `http://www.renater.fr`.

[22] Bernard M. Waxman. outing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399