Peer-to-Peer Replication with Preferences

C.Ye
Dept. of Information Engineering
The Chinese University of HK
Shatin, N.T., Hong Kong
cye5@ie.cuhk.edu.hk

D.M.Chiu
Dept. of Information Engineering
The Chinese University of HK
Shatin, N.T., Hong Kong
dmchiu@ie.cuhk.edu.hk

ABSTRACT

A P2P system can be viewed as a system that provides replication services. Unlike conventional structured replication systems (CDN, RAID), peers in an unstructured P2P system may have heterogeneous, sometimes low, online availability. Therefore, we formulate the problem with the objective to achieve good system level file availability, and study distributed algorithms for autonomous peers to accomplish that. In this paper, we emphasize the need to provide a differentiated replication service, since files are accessed with different frequency and have different importance. We quantify file preference in terms of weight and formulate the objective as to maximize a weighted sum of file availability. A bi-weight model is studied and then applied to a decentralized random replication algorithm through a statistical rounding policy. This algorithm is easily implementable by autonomous peers with partial information about the resources of the system, and yet yields favorable results in delivering the differentiated replication service while maintaining the system level replication goal.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms

Keywords

Peer-to-Peer, File Availability, Weight-based Replication

1. INTRODUCTION

No matter what the motivation might be, the autonomous peers in a peer-to-peer (P2P) system collectively accomplish a service. Our interest is in studying how the P2P system provides a replication service, and how the autonomous peers might be molded to provide a better replication service.

This question has been studied in [1], in which the authors considered how to replicate files among peers so as to minimize the bandwidth costs in accessing the files. They obtained an intuitively satisfactory answer that the amount of replication of each file should be proportional to the frequency of access to that file. Furthermore, this *proportional replication* service can be naturally implemented if each peer simply tries to manage its storage of files as a LRU (least recently used first) cache.

In [2], the authors considered a different system level metric to optimize - the average file availability. They allowed the peers to have different availability (proportion of time being online), and investigated several distributed algorithms for maximizing file availability - a random algorithm, a group-based algorithm and a greedy algorithm favoring using high availability peers first. Their conclusion indicates that each distributed algorithm can be favorable depending on whether variance of file availability is important, and whether simplicity or optimality is to be emphasized. In any case, as the amount of storage increases, the performance of the different algorithms also converge.

In this paper, we make a natural extension to [2]. We assume that the files may not be of equal importance, hence the desirability for their availability may also vary accordingly. We let the importance of the files be categorized by a weight distribution, and let the objective be the maximization of the weighted availability of the files. The weight of a file may be determined by different factors. In particular, it may be simply determined by the popularity (frequency of access) as in [1]. We call this differentiated replication service "*replication with preferences*" (or inter-changeably "*weight based replication*").

Collectively peers access some files more often than other files. If each time a file access is successful, it registers some (constant) benefit, then the total benefit over a long period of time is the sum of the weighted file availability where the weight is the frequency of access. In this sense, implementing a weight based replication service using frequency of access as weight treats all the files as equally important and it is a more natural objective than trying to give each file the same availability. More generally speaking, each successful access of a file may register a different amount of benefit depending on the file (or even the user). This can be modeled by additional weights that depend on the files (rather than the frequency of accessing them). The effect of putting more weights on some files over other files is to suitably shift more

INFOSCALE 2007, June 6-8, Suzhou, China

DOI 10.4108/infoscale.2007.927

storage resources to the files with higher weights, thus letting the P2P replication service achieve higher overall benefits. The weight-based models are clearly more powerful in representing real-life situations, even though the appropriate weights may not always be easily derivable.

In this paper, we first focus on effective algorithms to achieve weight-based replication assuming the weights (whether frequency of access or other file characteristics) are given. In this case, we find that files with bigger weight do receive more replication, but the amount of replication is not necessarily proportional to the weight, as in the case of [1]. This means we can no longer simply rely on the LRU caching algorithm to implement P2P replicaiton as in [1]. Fortunately, a simple distributed algorithm used by the peers can still deliver this *differentiated* replication service.

1.1 Related Works

Other than [1] and [2], there has been a considerable amount of research on object replication to different locations or computers. Most of the previous studies take the same approach of formulating the problem as an optimization problem and investigate suitable algorithms to achieve the optimal (or near-optimal) effectively.

In [3], the author studied the file allocation problem of replicating a single file over a set of computers. The file was demanded by various users who had different access costs to different computers. The author proved that the optimal assignment of the replicas to the computers that minimized the total cost was NP-complete. Authors in [4] studied the problem of replicating objects optimally in content distribution networks, in which each autonomous system (AS) replicated some objects and demanded objects stored in other ASes at the same time. The optimality of this replication was defined as the minimization of the average inter-AS hop distance a request must transverse. They proved that this optimization was NP-complete, and proposed some centralized heuristic algorithms to solve the replication problem.

The work in [5] focused on the problem of assigning file replicas according to the files' demand characteristics. The authors described a decentralized and adaptive protocol to replicate files in a way that balanced server load. Authors in [6] considered assigning replicas in an unstructured P2P system, focusing on minimizing the expected search size of the search queries. Their results showed that replicating files proportionally to the square root of a file's demand popularity is an optimal replication scheme. In [7], the authors proposed a suit of adaptive algorithms for replicating and replacing files as a function of evolving file popularity to maximize the "hit probability", i.e. the probability of finding the file within the community such as a LAN. Their algorithms were based on the structured p2p system, and assumed homogeneous peer online pattern.

Besides the above, web caching can also be regarded as a distributed replication network. In [8], the authors studied the problem of distributing a fixed set of web proxies in the Internet. There was a setup cost involved when assigning a proxy to a potential site, as well as a link cost for connecting any two potential sites. They modeled the network topologies as trees, and proposed a centralized algorithm to distribute the proxies with minimum total cost. Authors in [9] modeled this web sever placement problem as a K-median problem and developed several centralized placement algorithms to solve it.

As an orthogonal evaluation to the existing replication related studies, we put our focus on how to replicate a set of files so as to optimize file availability. There has already been some research dealing with this issue, such as [10, 11]. However, in our model, peers are allowed to have different online availability, which is more general in comparison with previous models [3, 4, 9, 10]. Our work differs from [11] in that [11] tried to achieve a pre-determined availability target while we are trying to optimize file availability based on given resources. Moreover, both [10] and [11] were equal weight replications.

1.2 Organization

In the rest of this paper, we first give the problem formulation; then solve the problem in a simple bi-weight case which is given by a closed-form solution, and discuss how the solution can be used recursively to extend the result to the multi-weight cases. Finally, we describe the simple distributed algorithm and use simulation to study its performance.

2. P2P REPLICATION SYSTEM

We first introduce notations and terminologies to formulate the problem. Table 1 summarizes the parameters used in our model.

Table 1: System parameters

$\mathcal{P}_i$	Writable peer set of peer i
$\mathcal{P}$	The set of peers in the system, $\mathcal{P} = \bigcup_i \mathcal{P}_i$
$\mathcal{F}_i$	The set of files to be replicated in peer i
$\mathcal{F}$	The set of files in the system, $\mathcal{F} = \bigcup_i \mathcal{F}_i$
M	Number of peers in the system: $M = \lvert\mathcal{P}\rvert$
N	Number of files in the system: $N = \lvert\mathcal{F}\rvert$
$\mathbf{p} = [p_i]$	Peer online availability vector
$\mathbf{s} = [s_i]$	Peer storage capacity
$\mathbf{w} = [w_j]$	File weight of each file j
$\mathbf{x} = [x_j]$	Replication redundancy of each file j
$\mathbf{n_H} = [n_{H(i)}]$	Number of high weight files in peer i
$\mathbf{n_L} = [n_{L(i)}]$	Number of low weight files in peer i
$\mathbf{R} = [r_{i,j}]$	A feasible replica placement
$\mathbf{p}[\mathbf{r}_j]$	Availability vector of peers replicating file j
$\mathbf{A} = [A_j]$	File availability distribution
$\mathbf{A_w}$	Overall weight based file availability

2.1 Peers

In our model, peers are assumed to cooperate for the same overall replication objective. When a peer joins the system, it is willing to offer a certain amount of storage for other peers to place their file replicas. In return, it can also distribute its file replicas to other peers, thereby increasing availability of its own files.

Each peer in this replication system is characterized by three parameters. First, we denote the online availability $p_i \in [0, 1]$ as the proportion of the time peer i stays online. When a peer is online, all the replicas it stores are assumed to be available and accessible by other peers in the system. Therefore, the probability of retrieving the replicas stored in peer i is equal to its availability p_i. Second, peer i has a set of files $\mathcal{F}_i$ that needs to be replicated. These files are of different weights. The third parameter is the amount of storage space that peer i offers for replication purposes, denoted by s_i. This shared storage is made available to other

peers in the system. We do not consider the bandwidth consumption between peers in this paper, and file replicas are assumed to be assigned to other peers in a negligible amount of time.

2.2 Estimation of File Availability

The availability of a file after replication is comprised of two parts: the replicas stored in the network and the original file in the peer who shares it, provided it is kept there. Due to storage limitation, it happens that some files cannot be replicated at all. Although users may be able to get access to an unreplicated file from the peer who shares it, provided the original copy is retained, the file in discussion is still considered to have 0 file availability in our model. That is to say, the contribution of the original copy is excluded. The reason behind this is to extract the file availability achieved by replication from the dependence on availability of the original copy. This promises a more explicit performance evaluation of the replication system. Moreover, peers may not always keep the files they share.

As a consequence, a file is available when at least one of its replicas is online. The probability of having at least one replica/peer available out of x is

$$P\{\geq 1 \text{ online}\} = 1 - (1-p_1)(1-p_2)\cdots(1-p_x). \quad (1)$$

Since each peer in the system may have a different online probability, and there may be a huge number of files involved, it is too expensive to calculate the exact availability for each file. Therefore, we use the average peer availability as an approximation:

$$A(\{p_i\}, x) \simeq 1 - (1-\bar{p_i})^x, \quad (2)$$

where $\bar{p_i}$ refers to the average peer availability of $\{p_i\}$. This approximation in fact gives a lower bound of the exact file availability.

2.3 Replication with Preferences

In our replication system, files are assigned with different weight values to indicate how important they are, and how they should be replicated. Our goal is to build a system in which highly weighted files receive relatively higher availability and vice versa. Such strategy is referred as *replication with preferences*. We define the overall system level availability as follows:

$$A_w = \frac{\sum_{j=1}^{N} w_j A_j}{\sum_{j=1}^{N} w_j}, \quad (3)$$

where w_j is the weight value of file$j$, and A_j is the resulted file availability of file$j$ after replication.

3. A WEIGHT BASED MODEL FOR P2P REPLICATION SYSTEM

A real world P2P replication system is complicated to model. Part of the problem comes from the complexity of network topology, partial connectivity and network dynamics. In this paper, we propose a simple weight based replication model to abstract away the above aspects. Although the model is simple, it is capable of illustrating the difficulties in resource allocation in a real world P2P replication system, specifically, heterogeneity of peer availability and file weight.

3.1 The Multi-Weight Replication Problem

Let us consider a replication system with a fixed population of $\mathcal{P}$ peers whose availability distribution is $\mathbf{p}$. Each peer i has a set of files $\mathcal{F}_i$ to be replicated. The set of all files to be replicated in the system is $\mathcal{F} = \cup_i \mathcal{F}_i$, with a weight distribution $\mathbf{w}$. We denote the number of peers as $M = |\mathcal{P}|$ and the number of files as $N = |\mathcal{F}|$.

When replicating, for a particular file $j \in \mathcal{F}_i$, peer i first needs to decide how much redundancy to add. This is denoted by the number of replicas $x_j(w_j)$, which depends on the file weight w_j. Then peer i creates x_j replicas, to be assigned to x_j different peers. For simplicity, it is assumed that all files in our model have the common size Γ, and the storage space offered by peer i is always in units of the file size.

From the angle of an individual file, x_j should be as large as possible in order to maximize file availability. The largest possible value is given by $x_j = M$. However, the total storage offered by peers is limited, so it is not always feasible for each file to be replicated by all peers. This implies the need of determining a suitable $x_j(w_j)$ for each file j from the angle of the overall replication system.

More generally, the problem is to seek a replica assignment policy. We define the replication placement matrix $\mathbf{R} = [r_{i,j}]_{M\times N}$:

$$\begin{matrix} r_{1,1} & r_{1,2} & \cdots & r_{1,N} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M,1} & r_{M,2} & \cdots & r_{M,N}, \end{matrix}$$

where $r_{i,j}$ indicates whether a replica of file j is assigned to peer i:

$$r_{i,j} = \begin{cases} 1 & : \text{ if peer } i \text{ stores a replica of file } j \\ 0 & : \text{ otherwise} \end{cases}$$

where

$$\begin{aligned} i &= 1, 2, \ldots M \\ j &= 1, 2, \ldots N. \end{aligned}$$

Obviously, peer i cannot store more than its storage capacity s_i:

$$\sum_{j=1}^{N} r_{i,j} \leq s_i \quad \forall i. \quad (4)$$

The number of replicas of file j stored in the system is equal to $x_j(w_j)$:

$$\sum_{i=1}^{M} r_{i,j} = x_j(w_j) \quad \forall j. \quad (5)$$

A replica placement matrix $\mathbf{R}$ is *feasible* only if it satisfies both conditions 4 and 5.

Let $\mathbf{r}_j$ denote the j^{th} column vector of the replica placement matrix $\mathbf{R}$, which then gives the subset of peers that replicate file j. We select the online availability of peers who replicate file j (i.e., $r_{i,j} = 1$), and denote it as $\mathbf{p}[\mathbf{r}_j]$. Then the overall weight based file availability can be computed as in equation 3:

$$A_w(\mathbf{R}) = \frac{\sum_{j=1}^{N} w_j A_j(\mathbf{p}[\mathbf{r}_j], x_j(w_j))}{\sum_{j=1}^{N} w_j},$$

where$A_j(\mathbf{p}[\mathbf{r}_j], x_j(w_j))$ is the availability of file j. Based on equation 2, we are able to rigorously define the resource allocation problem as a problem to find an optimal $\mathbf{R}$ to maximize the overall file availability $A_w(\mathbf{R})$.

3.2 The Bi-Weight Replication Model

The replication problem described in 3.1 actually includes two parts: storage allocation $\mathbf{x}$, and replica placement $\mathbf{R}$. To further simplify the problem, we now introduce the *bi-weight replication* model, which enables us to find an optimal resource allocation scheme among files.

In the bi-weight replication model, files are classified into two categories: high weight files(H-files) and low weight files(L-files), assigned with the weight value w_H and w_L respectively, where $w_H > w_L$. A file to be replicated in the system is either an H-file or an L-file. In fact, there are various metrics to determine the weight of a file, depending on the particular situation. For example, if we take popularity of files as the metric, the weight of a file can be determined by its accessing rate. To apply our model, a predetermined threshold is needed. If a file has an accessing rate higher than the threshold, it is assigned with w_H, otherwise, it is assigned with w_L. Specifically, we can consider the empirical file access patterns like Zipf distribution, which indicates that only a few files in a system are popular while many or most are, actually, less popular. Therefore, each peer can have certain percentage, e.g. 30%, of their files as H-files according to their accessing rates, and the remaining as L-files.

We intend to provide files in the same weight level with the same amount of storage resource. By equation 2 and 3, the problem can be formulated as

$$max: \frac{n_H w_H[1-(1-p)^{x_H}] + n_L w_L[1-(1-p)^{x_L}]}{n_H w_H + n_L w_L}, \quad (6)$$

where

$$n_H w_H + n_L w_L \leq s. \quad (7)$$

Here n_H and n_L are the number of H-files and L-files respectively, while x_H and x_L are the amount of redundancy per file for each weight level. s is the total storage space that offered by peers in the system, which gives a general constraint to the optimization problem.

In order to enhance the overall file availability, we assume a saturated usage of the storage resource by taking equality in constraint 7. We further denote:

$$q = 1 - p \in (0,1)$$

$$a_H = \frac{n_H w_H}{n_H w_H + n_L w_L}$$

$$a_L = \frac{n_L w_L}{n_H w_H + n_L w_L}$$

After using the constraint to reduce the variables, problem 6 becomes

$$max: 1 - (a_H q^{x_H} + a_L q^{\frac{s - n_H x_H}{n_L}}), \quad (8)$$

which is equivalent to

$$min: G = a_H q^{x_H} + a_L q^{\frac{s - n_H x_H}{n_L}}. \quad (9)$$

Since the function G is convex, and the gradient is

$$\nabla G = \ln q (a_H q^{x_H} - \frac{a_L n_H}{n_L} q^{\frac{s - n_H x_H}{n_L}}), \quad (10)$$

by letting $\nabla G = 0$, for $0 < w_L < w_H$ as defined, we then have

$$q^{\frac{(n_H + n_L)x_H - s}{n_L}} = \frac{a_L n_H}{a_H n_L} = \frac{w_L}{w_H} \in (0,1). \quad (11)$$

Since $q = 1 - p \in (0,1)$, this indicates

$$(n_H + n_L)x_H > s,$$

compared with 7, it follows that $x_H > x_L$. This is consistent with our expectation that provides the high weight files with more redundancy, which then naturally leads to higher file availability.

Based on the analysis above, the closed-form solution to problem 6 can be readily expressed as:

$$\begin{cases} x_H^* = \frac{s}{n_H + n_L} + \frac{n_L \frac{\ln \frac{w_L}{w_H}}{\ln(1-p)}}{n_H + n_L} \\ x_L^* = \frac{s}{n_H + n_L} - \frac{n_H \frac{\ln \frac{w_L}{w_H}}{\ln(1-p)}}{n_H + n_L} \end{cases}. \quad (12)$$

This solution indicates a logarithmic relationship between the amount of redundancy and system parameters such as file weight and file distribution pattern.

3.3 Discussion

In fact, the solution in equation 12 makes sense only when $s \geq n_H \frac{\ln \frac{w_L}{w_H}}{\ln(1-p)}$ because both x_H and x_L should be nonnegative. For the special situation $s < n_H \frac{\ln \frac{w_L}{w_H}}{\ln(1-p)}$, which means either that the number of H files in the system is extremely large or the L-files are of awfully low weight compared with the H-files. In both cases, the L-files are considered to be neglectable since they make trivial contributions to the system level file availability. Therefore, it degenerates to the equal weight replication, where $x = \frac{s}{n_H}$.

Although simplified, the bi-weight model is nontrivial. A straightforward extension is to further classify the H-files and/or L-files into two sub-categories, and then apply this model to each sub-storage allocation problem. Such extension enables the application of bi-weight model to multi-weight replication problems.

4. THE DISTRIBUTED REPLICATION ALGORITHM

By using the average peer availability as an approximation, the bi-weight model solves the problem of general storage allocation between H-files and L-files. We then need to find an optimal replica placement scheme $\mathbf{R}$ to assign the replicas properly in the network. In fact, similar problems were considered as combinatorial optimization problems in previous studies. They were proved to be NP-complete, and coupled with some heuristic solutions, as in [8, 9]. Such heuristic solutions can be implemented by a central agent that had all the necessary system parameters.

However, in a typical P2P system, there might be a huge number of peers whose participations are not synchronized, making timely collection of the system parameters from all peers intractable. Even if it is possible to collect all the parameters needed, it would be very time consuming for a central agent to solve this problem and distribute solutions to all other peers. Therefore, we focus on decentralized solutions that enable self operations by autonomous peers.

In this paper, we investigate a decentralized heuristic algorithm in which peers make their replication decisions independently to derive a feasible $\mathbf{R}$. These replication decisions are made based on the partial and limited information available to each peer, e.g. availability and storage space of the neighboring peers.

4.1 Writable peer set

P2P replication systems are constituted by connected peers. Unlike centralized or structured replication systems like RAID, peers in a P2P replication system may not be aware of the presence of all other peers in the system. We characterize the limited information available to each peer by introducing *degree of connectivity* for a peer. This is not the physical connectivity, but the logical reachability of a peer in terms of asking other peers to help it replicate a file. In this sense, a replication system with an indexing server, which allows each peer to know of all other peers' existence, can be considered as a replication system with 100% connectivity, despite the fact that peers are not directly connected to each other.

Given a degree of connectivity, a peer is called a *neighbor* of peer i if it can potentially be reached by peer i for replication. Due to storage limitation, it is possible that a peer cannot always distribute one replica to each of its neighbors. Hence, a peer i may choose a subset of its neighbors for replication. We name this subset the *writable peer set* $\mathcal{P}_i$. And we assume that no peer is left isolated in the system, therefore:

$$\mathcal{P} = \cup_i \mathcal{P}_i.$$

As described before, each peer requires several types of information from other peers in the writable peer set to facilitate making replication decisions. Such information can be encapsulated in the control protocol messages of a P2P system (such as the ping-pong messages in Gnutella), or can be transmitted in a separate protocol message. For each peer i' in the writable peer set of peer i, we define three types of information to be conveyed from i' to i.

1. The storage space offered by peer i' for replication, i.e. $s_{i'}$.
2. The number of files in different weight level that peer i' requests to replicate, i.e. $n_{H(i')}$ and $n_{L(i')}$, where $n_{H(i')} + n_{L(i')} = |\mathcal{F}_{i'}|$.
3. Online availability of peer i', i.e. $p_{i'}$.

4.2 The Statistical Rounding Policy

As discussed in section 3.1, a feasible replica placement solution must satisfy the storage constraint, namely, all replicas must fit into the storage space offered by the peers. To ensure this feasibility in a decentralized manner, each peer collects the pertinent information from its writable peer set and estimates suitable amount of storage overhead, i.e. x_H, x_L, for its files. In whole file replication, the number of replicas for any file should be an integer. However, this is not guaranteed in the bi-weight model. In order to channel the theoretical solution into applications, we introduce the *statistical rounding policy*.

We denote $ceil(x)$ as the minimal integer that larger than x, while $floor(x)$ the maximal integer that smaller than x. For a nonnegative x, both $ceil(x)$ and $floor(x)$ are nonnegative. The difference $ceil(x) - x$ is then defined as the *rounding threshold*. Specifically, to determine the exact number of replicas x_j that can be placed in the network by an H-file j, a random number $\lambda \in (0, 1)$ is generated. Then x_j is decided as the following:

$$x_j = \begin{cases} ceil(x_H) & : \text{ if } \lambda > ceil(x_H) - x_H \\ floor(x_H) & : \text{ otherwise} \end{cases}$$

The same method is also applied to L-files.

When determining the amount of replication redundancy x, there may be the situation that two peers having common elements in their writable sets replicate simultaneously, which would probably result in inaccurate estimation of the system parameters, e.g. the total storage so far available in the writable peer set. To avoid incorrect calculation of x, we employ the *locking phase* strategy, which works in the following way. Each peer in our system has a locking phase indicator to indicate its current state: locked or released. Once peer i begins the replication process (including estimation of x), all peers in its writable peer set $\mathcal{P}_i$ are set to be locked. While in the locking phase, a peer is "unaccessible" to other peers in the system except for peer i. These locked peers will be released when peer i finishes replication. Therefore, whenever a peer chooses the writable peer set, it must first remove the locked neighbors.

4.3 The Distributed Random Replication Algorithm

Generally, the availability of a file depends on (a) how many replicas can be placed in the network, which is measured in terms of the replication redundancy allocated to that file, and (b) the availability of those peers who store the replicas of that file. The bi-weight model tries to allocate the storage resource based on the weight of the files. The random replication algorithm, on the other hand, gives each file an equal opportunity in selecting peers. Both the processes are implemented by peers in a decentralized way. Hence, the entire replication algorithm includes the following two steps.

The distributed replication algorithm

Writable peer set estimation:

1. Peer i chooses the writable peer set $\mathcal{P}_i$.
2. All peers in $\mathcal{P}_i$ are "locked".
3. Estimate $S_i = \sum_{i' \in \mathcal{P}_i} s_{i'}$
   $N_{H(i)} = \sum_{i' \in \mathcal{P}_i} n_H(i')$
   $N_{L(i)} = \sum_{i' \in \mathcal{P}_i} n_L(i')$
   $\bar{p_i} = \frac{\sum_{i' \in \mathcal{P}_i} p_{i'}}{N_{H(i)} + N_{H(i)}}$.
4. Estimate $\{x_{H(i)}, x_{L(i)}\}$.

To replicate file j:

5. Generate a random $\lambda \in (0, 1)$ and determine x_j by the statistical rounding.
6. Create x_j replicas of file j.
7. IF peer i cannot find x_j peers with available storage space, skip replicating this file.
8. ELSE peer i randomly picks x_j peers from $\mathcal{P}_i$ to store the replicas. Update available storage space of these x_j peers.
9. Peers in $\mathcal{P}_i$ are released.

Storage allocation: First, each peer i calculates the average peer availability ($\bar{p_i}$), the total storage space offered

Table 2: simulation setups

Expected number of files per peer	File weight pattern	Percentage of H-files
Uniform in [0,20] / Expected 10	$w_H : w_L = 2 : 1$	[0, 100%]
Uniform in [0,100] / Expected 50	$w_H : w_L = 2 : 1$	[0, 100%]
Uniform in [0,200] / Expected 100	$w_H : w_L = 2 : 1$	[0, 100%]
Uniform in [0,100] / Expected 50	$w_H : w_L = 4 : 1$	[0, 100%]

(S_i), and the total number of files of each weight level to be replicated, namely, $\{N_{H(i)}, N_{L(i)}\}$, by peers in its writable peer set. Peer i then estimates the replication redundancy for all files $j \in \mathcal{F}_i$ by:

$$\begin{cases} x_{H(i)} = \frac{S_i}{N_{H(i)}+N_{L(i)}} + \frac{N_{L(i)} \frac{\ln \frac{w_L}{w_H}}{\ln(1-p_i)}}{N_{H(i)}+N_{L(i)}} \\ x_{L(i)} = \frac{S_i}{N_{H(i)}+N_{L(i)}} - \frac{N_{H(i)} \frac{\ln \frac{w_L}{w_H}}{\ln(1-p_i)}}{N_{H(i)}+N_{L(i)}} \end{cases}.$$

This replication redundancy estimates how much storage space each file (in the writable peer set) can use on average. If all peers in the system cooperate and follow such estimation, it is very likely that the storage space will not be overused.

Replica placement: Before replicating file j, peer i applies the statistical rounding policy to determine x_j, the exact number of replicas of file j, based on $\{x_{H(i)}, x_{L(i)}\}$ from last step and the actual weight of file j. After that, peer i randomly picks x_j peers, whose storage space is not exhausted, in its writable peer set $\mathcal{P}_i$. Replicas of file j are then produced and assigned to these peers. Peer i stops the replication process when all its files are replicated, or when storage space in the writable peer set runs out.

To enhance the overall weight based file availability of the system, the high weight files are entitled with priority in the order of replication. That is, the high weight files in the system will be replicated before the low weight files.

5. EVALUATION OF THE SYSTEM

The performance of the replication algorithm is evaluated by simulations. The concerned file availability is used as the main metric, including a) the weight based system level file availability $E[A_w]$ which measures how the peers replicate at the overall replication goal, and b) the actual expected file availability $E[A_H]$ and $E[A_L]$ which measures how the files in the system are treated differently based on their weights. In addition, we employ the variance $Var[A_H]$ and $Var[A_L]$ as a fairness measurement of the achieved file availability in the same weight level.

We first discuss the simulation setups, followed by the results.

5.1 Simulation Setup

We simulate a replication system with 100 peers that are randomly linked. The connectivity of the network is controlled by a parameter $m \in [0, 1]$. Any two peers in the system are linked if a uniformly generated random number in $[0, 1]$ is greater than m. So the expected number of links would be $(1 - m)N(N - 1)/2 = 4950(1 - m)$. These links are logical, and the link delays and transmission costs are ignored in our model. In fact, the parameter $(1 - m)$ can be viewed as the degree of connectivity of the system.

We are interested in how the algorithm perform under different system parameters, such as the total number of files to be replicated in the system, the weight patterns, and the percentage of high weight files out of the general file set. These parameters are summarized in table 2. In all simulations, the system has a degree of connectivity of 0.5, and the peers' online availability is uniformly distributed in $(0, 1)$. The system-wide *average replication redundancy*, defined as the ratio of the total storage space offered by all peers to the total size of all files in the system regardless of file weight, is set to be

$$E(\hat{X}) = \frac{\sum_{i \in \mathcal{P}} s_i}{\sum_{j \in \mathcal{F}} f_j} = 2.$$

That means, each file, no matter with a high weight or a low weight, is expected to have 2 replicas stored in the network on average. However, the exact storage space offered by each peer is not the same. Instead, it followed a uniform distribution with $E(\hat{X})$ as the expected value. In fact, variations in these parameters, i.e., the online pattern of peers, the degree of connectivity and the average replication redundancy, can also result in direct impact on file availability. This has been studied in our previous work [2].

To minimize simulation errors due to random perturbations, each simulation setup is run 200 times, and the average results are reported.

5.2 Simulation Results

We first evaluate the system performance in terms of the overall file availability. The theoretical results generated by the bi-weight model under the same expected parameters are used as the benchmark for evaluation.

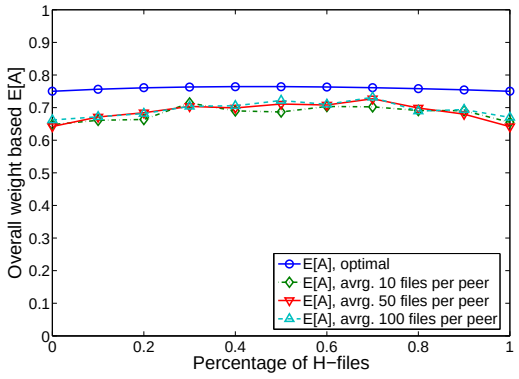

Figure 1: System level weight based file availability, $w_H : w_L = 2 : 1$, avrg. $\{10, 50, 100\}$ files per peer

Figure 1 shows the results of the system level weight based file availability, against the percentage of high weight files

out of all the files to be replicated. The expected total number of files varies from 1000 to 10000. Compared with the optimal solution, the file availability achieved by simulations is 2% – 10% inferior. As the number of files to be replicated in the system increases, the achieved file availability tends to approach the optimal, but the growth is not significant. The reasons for such a gap between the achievable and the optimal may come from the following: (a) The total number of files and the percentage of H-files are precise in the analysis. However, these parameters in the simulations are random, generated by using the exact numerical values in the analysis as the mean. (b) Peers only have partial and limited information of the system resource from their neighbors, which may result in inconsistent estimations among peers depending on a particular writable peer set. (c) The statistical rounding policy also introduces certain amount of randomness.

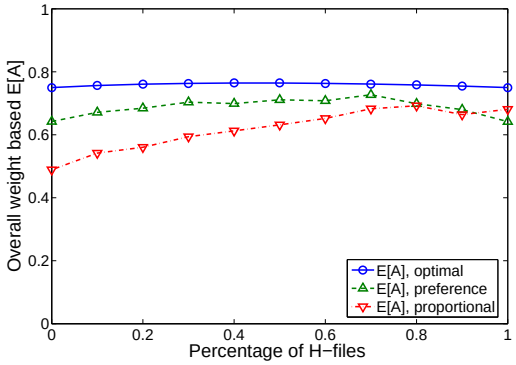

Figure 2: System level weight based file availability, $w_H : w_L = 2 : 1$, avrg. 50 files per peer

A comparison of differentiated replication and proportional replication is shown in figure 2. The result reveals that when the metric shifts to the overall file availability, the proportional relationship between the number of replicas and file weights can no longer achieve optimality.

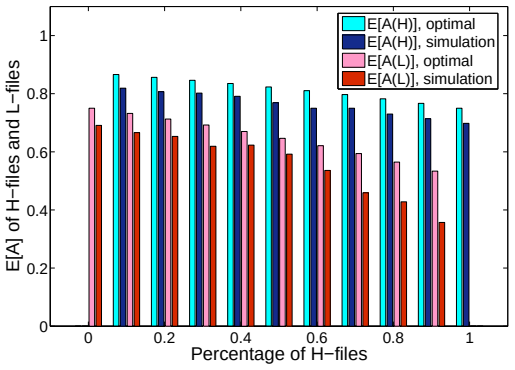

Figure 3: Achieved file availability of H-files and L-files, avrg. 50 files per peer, $w_H : w_L = 2 : 1$

As a measurement of the differentiated replication service, figure 3 shows the actual file availability of the high weight

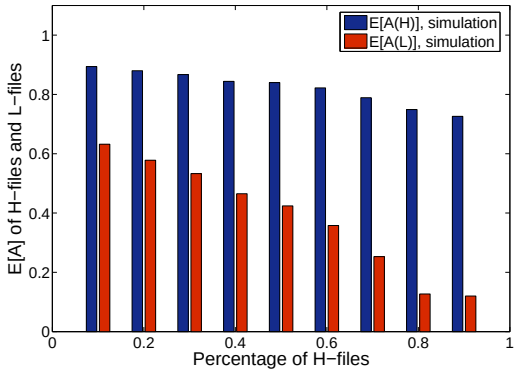

Figure 4: Achieved file availability of H-files and L-files, avrg. 50 files per peer, $w_H : w_L = 4 : 1$

and low weight files respectively. It can be observed that when the percentage of H-files is low, e.g. 10%, they can receive an availability up to around 85% after replication. As the percentage increases, the availability of H-files decreases. The explanation is that when the percentage of H-files turns high, hence the number of such files becomes large, the H-files turns to be "less important" compared to previously. When the percentage of H-files is 0% or 100%, the replication degenerates to equal weight replication.

However, the H-files always get higher availability than the L-files, regardless of their percentage. This meets the goal of the differentiated replication service. Moreover, when the difference of weights grows, e.g. from $w_H : w_L = 2 : 1$ to $w_H : w_L = 4 : 1$, the availability of H-files and L-files diverges further. The comparison of figure 3 and 4 provides an illustration of this.

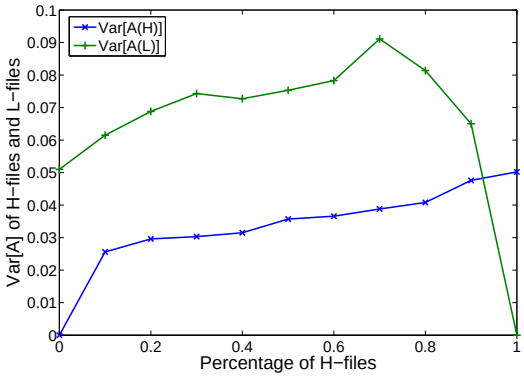

Figure 5: Variance of achieved file availability of H-files and L-files, avrg. 50 files per peer, $w_H : w_L = 2 : 1$

To gain a more comprehensive understanding of our model, we evaluate the variance of the achieved file availability and the portion of files that is successfully replicated. It can be observed from figure 5 and figure 6 that the $Var[A]$ of H-files is significant lower than that of L files. However, when the percentage of H-files exceeds a certain threshold, i.e. $\geq 70\%$ in our simulation, $Var[A_L]$ turns down. This is because when the H-files become the majority in file population, some L-files are left unreplicated due to the storage

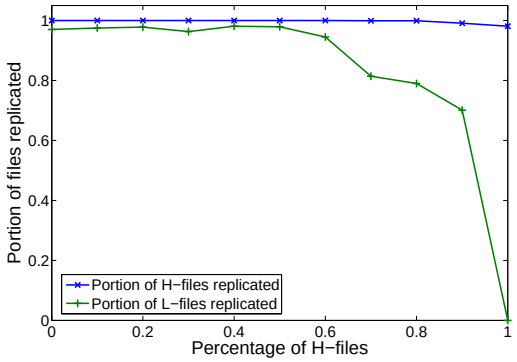

Figure 6: Portion of H-files and L-files that be successfully replicated, avrg. 50 files per peer, $w_H : w_L = 2 : 1$

limitation. (They are considered to have 0 availability in the computation of achieved objective). Most often, the H-files can be 100% replicated while near 100% of the L-files are replicated, especially when the H-files are rare in the system. The above results reveal that weight based replication tends to guarantee the service quality in favor of the high weight files while trying to maintain that of the low weigh files at a reasonable level.

6. DISCUSSION AND FUTURE WORK

The simulation results indicate that our replication algorithm delivers a respectable replication service with preferences while maintaining a good system level file availability. Yet there are still many interesting topics for future work.

For example, as alluded to in the introduction, we assume the file weights are given. A crude system is to let the peers set the file weights themselves. This assumes the peers are honest and shares an established framework for setting file weights. In real life systems, such a crude system would not be dependable as the peers would have selfish reasons to not be truthful. Instead, it is necessary to include an incentive system to make the peers set the weights truthfully. One possible consideration is to tie a peer's contribution (storage and online time) to the weights it gives to files. Obviously, there are many interesting mechanisms one can design, together with the system service model.

Another extension is related to network dynamics. In this paper, we assume a *static* replication system: a fixed set of peers join the system and each of them replicates a fixed set of files. In a real world system, peers continuously join and leave; they may remove old files and introduce new files; the preference of a file may change over time. These dynamics bring many interesting possibilities. For example, if some peers leave permanently, the availability of the files it stored will be affected. How do we redistribute these file replicas in order to maintain the file availability in this scenario?

These problems are all worthy of future study.

7. CONCLUSION

In this paper, we considered the design of weight-based replication services in unstructured peer-to-peer systems. We represent file preferences by weights, and formulate the resource allocation problem as an optimization problem that maximizes the overall weight based file availability.

Since the problem is computation-intensive, we investigate a bi-weight model, in which the optimal solution reveals a logarithmic relationship between the amount of replication and system parameters like file weight and file distribution patterns. Though simple, the bi-weight model can be extended to multi weight cases by recursion. We then apply the solution to a distributed random replication algorithm by introducing a statistical rounding policy. The algorithm is implemented by autonomous peers who cooperate with the neighbors, based on partial and limited information about system resources. The performance of this algorithm under different system parameters is evaluated by simulation. Our results show that, by employing the analytical results for the optimal solution, the distributed algorithm yields favorable performance in delivering the differentiated replication service while achieving a good system level file availability.

8. ACKNOWLEDGMENTS

This work is partly supported by Hong Kong RGC 411505, and is affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.

9. REFERENCES

[1] S. Tewari, and L. Kleinrock. Proportional Replication in Peer-to-Peer Networks, in Proc. of *Infocom* 2006.

[2] W. K. Lin, C. Ye, and D. M. Chiu. Decentralized Replication Algorithms for Improving File Availability in P2P Networks, to appear in *the 15th IEEE International Workshop on Quality of Service*, 2007.

[3] K. P. Eswaran. Placement of records of a file and file allocation in a computer network, in Proc. of *IFIP Conference*, 1974.

[4] J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks, in Proc. of *the 6th International Workshop on Web Content Caching and Distribution*, 2001.

[5] V. Gopalakrishnan, B. Bhattacharjee, and P. Keleher. Adaptive replication in peer-to-peer systems, in Proc. of *The 24th International Conference on Distributed Computing Systems*, 2004.

[6] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks, in Proc. of *ACM SIGCOMM* 2002.

[7] J. Kangasharju, K. W. Ross, and D. A. Turner. Adaptive Content Management in Structured P2P Communities, in Proc. of *The First International Conference on Scalable Information Systems*, 2006.

[8] M. J. G. Bo Li, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the Internet, in Proc. of *Infocom* 1999.

[9] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas, in Proc. of *Infocom* 2001.

[10] K. Ranganathan, A. Iamnitchi, and I. Foster. Improving data availability through dynamic model-Driven replication in large peer-to-peer communities, in Proc. of *The 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[11] F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen. Autonomous repliaction for high avaibility in unstructured P2P systems, in Proc. of *The 22nd International Symposium on Reliable Distributed Systems*, 2003.

[12] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs, in Proc. of *Sigmetrics* 2000.