SCIENCE CHINA
Information Sciences

• **RESEARCH PAPERS** •

August 2011 Vol. 54 No. 8: 1585–1595
doi: 10.1007/s11432-011-4313-9

User-experience-based availability analysis model and its application in P2P storage systems

WU Yu*, YANG Zhi, QU Zhi, XIAO Zhen & DAI YaFei

Department of Computer Science, Peking University, Beijing 100871*, China*

Received October 7, 2009; accepted November 30, 2010

Abstract Data availability is one of the most important properties of peer-to-peer (P2P) storage systems. Availability analysis model and data placement are two key design choices. Users in P2P storage system are both providers and customers. This characteristic determines that the availability analysis must be user-centric, and thereby enhance the quality of service and decrease the system cost. The popular approach in recent studies is simple random placement with steady-state model, which has the following drawbacks: 1) It ignores the up/down patterns of nodes, whose availability is over-estimated or under-estimated at different periods of time. 2) It ignores the access patterns of users, so the availability perceived by users is hard to evaluate precisely. 3) It ignores the huge difference of nodes' availability, thus leading to the absence of incentive. This paper proposes a novel user-experience-based availability model, which evaluates the availability of P2P storage system in terms of user experience, which can degenerate to traditional availability analysis model. Based on the new model, this paper proposes decentralized data placement algorithms for two typical P2P storage applications: "data sharing" and "personal backup". By the trace-driven simulation, we prove that our methods can enhance the availability perceived by users greatly, reduce the variance of the availability dramatically and eliminate the nodes with low availability in data-sharing applications; meanwhile, it can provide different-level service to encourage users according to their contributions.

Keywords P2P storage, user experience, availability, data replacement

Citation Wu Y, Yang Z, Qu Z, et al. User-experience-based availability analysis model and its application in P2P storage systems. Sci China Inf Sci, 2011, 54: 1585–1595, doi: 10.1007/s11432-011-4313-9

1 Introduction

Availability is a storage system property which measures the probability that data can be retrieved. P2P network provides favorable conditions for storage system because of its high scalability and autonomy. Therefore, the key design issue in P2P storage system is to ensure high availability under users' high dynamics. In many applications, not all resources are required to be up at every point in time. A resource is required to be up only during the time periods when the user requests it. Therefore, the data placement aimed at appreciated availability is more helpful for rational allocating and saving resources. The challenges of user-based availability analysis are 1) How to describe the users up/down behaviors accurately, 2) how to evaluate the data availability from users' perspective, 3) how to choose peers to place replications to enhance users' appreciated availability.

*Corresponding author (email: viewrain@gmail.com)

© Science China Press and Springer-Verlag Berlin Heidelberg 2011

The current study [1–5] has used the steady-state availability analysis model and random placement. This approach has three drawbacks: 1) It ignores the up/down patterns of nodes, whose availability is over-estimated or under-estimated at different periods of time, so it cannot describe the users up/down behaviors accurately. 2) It ignores the access patterns of users in different P2P storage applications, so the availability perceived by users is hard to evaluate precisely. 3) It ignores the huge difference of the nodes' availabilities which present their contribution, so that the resource allocation usually does not match the contribution, thus leading to the absence of incentive.

This paper constructs an availability analysis model from the users' standpoint based on describing the up/down pattern and availability of nodes accurately, provides decentralized data placement algorithms for two typical P2P storage applications respectively, solves the three main issues mentioned before, and proves that the algorithms can enhance the user experience effectively and effect the whole system positively.

2 Related work

In the existent P2P-based network storage systems, such as TotalRecall [1], Farsite [6], and OceanStore [7], there have been many studies in data placement. But these algorithms do not take the up/down patterns and contributions of users into consideration. In contrast, this paper brings up the availability analysis from the users' standpoint and proposes the corresponding data placement algorithms.

Some research allows for the diversity of nodes and files. The study [8] targets to place the popular files on the high availability nodes for enhancing the availability of system service. The study [9] proposes several heuristic algorithms including the greedy algorithm and the group partition algorithm with the assumption that the access frequency of each file is the same. However, they do not consider the time factor. In contrast, this paper considered the behaviors of users in different time and their impact on the availability, then optimizes the user perceived availability in another dimension.

Some recent studies [10, 11] allow for the time factor, but they have different perspectives. Kim et al. [10] provided the time-related replication algorithm, which predicts the offline time according to the length of sessions before. Tian et al. [11] proposed a fine-gained stochastic process model, focusing on the data availability changed with time. These models have not described the users' up/down pattern in a long period. Instead, this paper focused on finding out the patterns, constructing the availability analysis model from the users' point of view, and rationally placing files with available node resources to optimize the user experience.

3 User experience-based availability analysis model

In this section we introduce how to describe the data availability from the perspective of the users. First, we introduce the P2P storage system and the corresponding method of measurement. Second, we provide the way to describe the nodes' pattern accurately. Third, we describe the data availability using vector. At last, by integrating the availability analysis model with the access pattern, we construct the user experience-based availability analysis model.

3.1 P2P storage system

This paper focuses on the P2P storage system with a large number of peers, which cooperate to replicate files for each other. When a peer joins the system, it can offer a certain amount of storage resources for other peers to place their file replicas. In return, it can also distribute its file replicas to other peers. Table 1 summarizes the parameters used in our model.

3.2 The patterns of peers' up/down behaviors

It is very important to accurately describe the peers' up/down behaviors for analyzing the user-centric availability. On one hand, the files' availability is determined by the up/down behaviors of the peers who

Table 1 System parameters

Parameter	Description
F_i	The set of files to be replicated in peer i
F	The set of files in the system: $F = \bigcup_i F_i$
P_i	Writable peer set of peer i
P	The set of peers in the system: $P = \bigcup_i P_i$
N	Number of peers in the system: $N = \|P\|$
M	Number of files in the system: $M = \|F\|$
$p_i = \langle p_{i1}, \ldots, p_{in} \rangle$	Availability vector of peer i, $p_{it} \in [0, 1]$ presents the availability of peer i at the tth time slot
s_i	The storage space provided by peer i
f_j	File size of file j
n_j	Number of replicas for file j, not including the original one provided by the owner.
$R = [r_{ij}]$	A feasible replica placement
$p = [r_j]$	Availability vector of peers replicating file j
$A_j = \langle A_{j1}, \ldots, A_{jn} \rangle$	File availability vector of file j, $A_{jt} \in [0, 1]$ presents the availability of file j at the tth time slot
$p[d_j]$	Access-probability vector of the peers requesting file j, we use their availability vector for simplicity
$U = [U_{ij}]$	The user perceived availability, U_{ij} presents the availability of file j perceived by peer i

replicate the files. On the other hand, the users' access patterns can affect the user perceived availability.

Ref. [12] shows that a peer has regular behavior in the period of one day in high dynamic P2P systems. So we use the availability vector $p_i = \langle p_{i1}, \ldots, p_{in} \rangle$ instead of the average availability. The calculation method is as follows: In the certain period of time t, count the time t_{online} when the peer is online, so its availability p in this period is

$$p = t_{\text{online}}/t. \tag{1}$$

Since the peers' regular behaviors, we investigate in the availability at different time slot in one period T (one day). Assuming that the availability in Δt is stable, then we get the availability vector of peer i by (2):

$$p_i = \langle p_{i1}, \ldots, p_{in} \rangle, \quad n = T/\Delta t, \quad p_{it} \in [0, 1]. \tag{2}$$

p_{it} presents the availability of peer i at Δt_t. Gao et al. [13] has proved that the peers' up/down states are similar at the same time slot every day, so the vector can describe the peers' up/down pattern accurately; besides, the average of vector components is the traditional system availability, so the availability vector can be used to describe the up/down pattern as well as the overall availability. We cluster the peers' patterns using the trace of MAZE system (subsection 5.1), and get some conclusions: 1) A peer's availability varies at different time slot, so the average value cannot show the peer's characteristic completely; 2) peers can have different up/down patterns, even if they have the same average availability; 3) the active time of the peers in different clusters has 2–7 h drift, so the availability can be complementary; 4) the most available peers always keep up, they can offer more positive impact if we can split them and collocate with the peers with low availability.

3.3 Data availability

Assuming that file j is replicated on k peers, $\{p_i\}$ presents the availability vectors of the k peers, the availability of file j at the tth time slot is A_{jt}:

$$A_{jt} = P(\text{at least one peer is online}) = 1 - P(\text{none of peers is online}) = 1 - \prod_{i=1,\ldots,k} (1 - p_{it}). \tag{3}$$

So we get the data availability:

$$A_j = A_j(\{p_i\}) = \left\langle 1 - \prod_{i=1,\ldots,k} (1 - p_{i1}), \ldots, 1 - \prod_{i=1,\ldots,k} (1 - p_{in}) \right\rangle. \tag{4}$$

The number of peers and the shared storage space in the system is limited, so the maximum redundancy of the file is $r^* = \sum_{i=1}^{N} s_i / \sum_{j=1}^{M} f_j$, assuming that each file has the same replicas, so $n_j \leqslant r^*$. When replicating, we choose at most n_j available peers to place the data, thus we get the data placement matrix $R = [r_{i,j}]_{N \times M}$, in which $r_{i,j}$ presents that if file j is distributed on peer i or not:

$$r_{i,j} = \begin{cases} 1, & \text{if peer } i \text{ store } f_j, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, M.$$

For each peer, the size of replicated files should be less than its shared space size, so we get (5); for each file, the number of replicas can be calculated by (6):

$$\forall i, \sum_{j=1}^{M} r_{i,j} f_j \leqslant s_i, \tag{5}$$

$$\forall j, \sum_{i=1}^{N} r_{i,j} = n_j \leqslant \sum_{i=1}^{N} s_i / \sum_{j=1}^{M} f_j. \tag{6}$$

r_j presents the jth column of R, where we can get the peer set to store file j, we use $p[r_j]$ to present the availability vectors of the peer set, so the data availability vector of file j is $A_j = A_j(p[r_j])$, which can be calculated from (3) and (4).

3.4 User-experience-based availability analysis

Subsection 3.3 analyzes the availability at different time from the standpoint of data. However, for users, different access modes will lead to different user experience. In this section, we construct the availability analysis model from the standpoint of users.

$U = [U_{ij}]$, U_{ij} presents the availability of file j perceived by peer i. Assuming the availability vector of file j is $\langle A_{j1}, \ldots, A_{jn} \rangle$ and the availability vector of peer i is $\langle p_{i1}, \ldots, p_{in} \rangle$. We use the peer's availability vector in place of its access vector for simplicity, which means the peer may access the files at any time when it is online.

Thus the availability of file j perceived by peer i is

$$U_{ij} = \sum_{t=1}^{n} p_{it} A_{jt} / \sum_{t=1}^{n} p_{it}. \tag{7}$$

For each peer, the denominator in formula (7) is determined by itself, so our target is to maximize $\sum_{t=1}^{n} p_{it} A_{jt}$, under the limitation of existing resources, in order to enhance U_{ij}. Formula (7) shows that the access probability of the file is related to the availability vectors $p[d_j]$ of the accessors. If file j is only requested by peer i, the p_{it} in (7) is only related to the availability vector of this peer. If file j is requested by k users, the p_{it} in (7) is the average of the corresponding vector component of the k users.

4 Heuristic data placement algorithms

In this section, we introduce three heuristic data placement algorithms: 1) existent random data placement algorithm, 2) "Different Levels and Pattern-complemented" placement algorithm, 3) "Same Level and Pattern-similar" placement algorithm.

4.1 Preliminary knowledge

4.1.1 *Information collection*

Ref. [9] described 1) the set of Writable peers P_i, which is the set of the neighbor peers can be used to store data by peer i, 2) estimation of the maximum redundancy: use the locking phase strategy to avoid multiple counting the peers' resources, and then calculate the maximum number of replicas n_i by formulas (8)–(10):

$$S_i = \sum_{i' \in P_i} s_{i'}, \tag{8}$$

$$Z_i = \sum_{i' \in P_i} \sum_{j \in F_{i'}} f_j, \tag{9}$$

$$n_i = S_i / Z_i, \tag{10}$$

3) message passing: More information should be added into the protocol message: the storage space $s_{i'}$ of peer i', the size of the files needed to be distributed and the up/down information of itself. In this section, we reuse the definitions, but the difference is that in [9] it passes the message of average availability, not the up/down pattern message.

4.1.2 *The messages of up/down pattern and data availability*

When designing a real system, we use the bit string to present the up/down patterns. Each peer i traces the up/down statuses in the recent period and records it as a bit string, for example, 1 presents online status, 0 presents offline status, b_i is like 101111001010. We use $\text{len}(b_i)$ to present the length of the string, use $\{b_i\}$ to present the number of 1 in the string, then the average availability of peer i is

$$A_i = \{b_i\}/\text{len}(b_i). \tag{11}$$

We can also count the time slots of co-occurrence and the union set of the up time slots. Assume that the pattern bit strings of peer A and peer B are b_A and b_B respectively, then

Time slots # of co-occurrence: $C_{AB} = \{\text{len}(b_A \& b_B)\}$,

Up time slots # of union set: $O_{AB} = \{\text{len}(b_A | b_B)\}$,

Co-occurrence factor: $\rho = C_{AB}/O_{AB}$,

Coverage factor: $\sigma = O_{AB}/\text{len}(\{b_A\} + \{b_B\})$.

Apparently, the bigger the ρ is, the more the similarities of their availabilities are; the bigger the σ is, the larger the differences of their availabilities and/or their up/down patterns are, which means they are more complementary. The data availability is calculated by OR. For example, file j has three replicas, replicated on peer A, B and C, assuming that their pattern bit strings are b_A, b_B and b_C, then the bit string of file j is $A_j^b = b_A | b_B | b_C$. By this bit string, we can get the availability vector of the data as well as the average availability.

4.2 Random placement algorithm

There are two steps in random placement algorithm. First, each peer i collects the messages of the writable peers and calculates the number of feasible replicas n_i by formulas (8)–(10), then randomly chooses n_i peers from the writable peers set. This is introduced in [9], so we do not go into the details here.

4.3 "Different Levels and Pattern-complemented" placement algorithm

"Different Levels and Pattern-complemented" placement algorithm is for the popular shared files, which has two targets: 1) allocating the resources to each popular file fairly; 2) the coverage of the up time of all the storage peers is as wide as possible, so that the popular shared files can be accessed by users at different time. The algorithm includes two steps:

Table 2 "Different Levels and Pattern-complemented" placement algorithm

Evaluation of the writable peer set:

1. peer i chooses the writable per set P_i.
2. locks all the peer $x \in P_i$.
3. Evaluates $S_i = \sum_{i' \in P_i} s_{i'}$ and $Z_i = \sum_{i' \in P_i} \sum_{j \in F_{i'}} f_j$, calculates $n_i = S_i/Z_i$.

Replica placement:

4. for each $x \in P_i$: calculates the average availability A_x.
5. sorts $\{A_x\}$, $x \in P_i$, partitions the ascending sorted peers into L groups: $g = \{g_1, g_2, \ldots, g_L\}$.
6. for each $f_j \in F_i$: randomly selects a peer $x \in P_i$ to place the first replica, initializes (A_j^b).
7. $r = 2$ to n_i: // places the rest $n_i - 1$ replicas.
   - 7.1 for each f_j in F_i: calculates A_j by A_j^b.
   - 7.2 F_i' = sorts $\{f_j\}$ by A_j, $f_j \in F_i$.
   - 7.3 for each $f_j \in F_i'$:
     - 7.3.1 calculates A_j and the interval l it belongs to.
     - 7.3.2 selects a peer $x \in g_{L+1-l}$(has enough space) to store the rth replica, maximize $\sigma(b_x, A_j^b)$.
       - i. if $x = \phi$, selects peer x in the neighbor group of g_{L+1-l}, maximize $\sigma(b_x, A_j^b)$.
       - ii. if $x = \phi$, skips.
     - 7.3.3 $A_j^b = A_j^b \& b_x$.
8. releases the peers in P_i

1) Replicas calculation: The same as it for random placement algorithm.

2) Replica placement: Calculate the average availability for each peer in P_i, sort them in ascending order, partition the ordered peer set into L groups: $g = g_1, g_2, \ldots, g_L$. We can get L intervals: $[min_1, min_2), [min_2, min_3), \ldots, [min_L, +\infty)$, in which min_i is the smallest value in g_i. If a peer's availability is in the lth interval, the peer belongs to the lth level. For each file j owned by peer i, first randomly selects a peer (who has available storage space) to store the first replica, calculates the current file's availability bit string A_j^b; then circularly distributes the rest replicas: 1) calculates the average availability of every file according to A_j^b, sorts them in ascending order, then gets $\langle A_1^b, A_2^b, \ldots, A_k^b \rangle$ (assuming that peer i has k files to store); 2) for each file j (in the sorted order), evaluates the availability and its level l, then selects a peer in g_{L+1-l} to store the next replica, which has the largest coverage factor with and has enough space. If there are no available peers in g_{L+1-l}, select the peer in the neighbor group. Here we omit this case; 3) update A_j^b. The detail of the algorithm is shown in Table 2.

4.4 "Same Level and Pattern-similar" placement algorithm

"Same Level and Pattern-similar" placement algorithm is for the personal backup system, the targets are 1) as far as possible to ensure the requested data's availability when the user is up, so place the data in the neighbor peers who have the similar up/down patterns; 2) to encourage the users to provide resources, making the perceived data availability have positive correlation with the users' own availability. The algorithm is shown in Table 3, including two steps:

1) Replicas calculation: The same as it for random placement algorithm.

2) Replica placement: First, calculate the average availabilities of each peer in P_i, sort them in ascending order, partition the ordered peer set into L groups: $g = g_1, g_2, \ldots, g_L$. For each peer i in group l, for each file j, choose the peers 1) with the maximum occurrence factor, 2) in the same group or neighbor group, 3) have enough space to store it.

5 Evaluation

In this section, we use trace-driven simulation 1) to study the user perceived difference of availabilities when using traditional random placement algorithms and using our new algorithms (subsections 4.2 and 4.4), and 2) to compare the performance in P2P systems with different connectivity and different redundancies.

Table 3 "Same Level and Pattern-similar" placement algorithm

Evaluation of the writable peer set:

1. peer i chooses the writable per set P_i.
2. locks all the peer $x \in P_i$.
3. Evaluates $S_i = \sum_{i' \in P_i} s_{i'}$ and $Z_i = \sum_{i' \in P_i} \sum_{j \in F_{i'}} f_j$, calculates $n_i = S_i / Z_i$.

Replica placement:

4. for each $x \in P_i$: calculates the average availability A_x.
5. sorts $\{A_x\}$, $x \in P_i$, partitions the ascending sorted peers into L groups: $g = \{g_1, g_2, ..., g_L\}$.
6. for each $f_j \in F_i'$:
   - 6.1 if: $f_j \in g_l, |g_l| \geqslant n_i$, chooses n_i peers X to store replicas, maximize ρ.
   - 6.2 else:
     - 6.2.1 $X = g_l$, stores $|g_l|$ replicas on them.
     - 6.2.2 chooses the peers in the neighbor group of g_l, maximize ρ.
7. releases the peer in P_i

5.1 Trace data

In this paper, the simulation based on the real trace of MAZE (http://maze.pku.edu.cn), which is one of the largest non-commercial P2P file-sharing system over CERNET (China Education and Research Network), with more than two million registered users and an average of 20K simultaneous online users, developed by our research group. As said in [14], the peers in this system are with high dynamics, low mean availability (0.26) and limited bandwidth. Our previous studies [14–16] have demonstrated that the traffic and user behaviors in MAZE are similar to that in many other P2P file-sharing systems. Therefore, our analysis is applicable for them.

The online log of Maze is a snapshot of the total online users' information every three minutes so we can know that when a peer joins and leaves the system. If one peer was not online for the last two weeks, it is supposed to leave the system permanently. We collect the daily log of MAZE during one month period March 1–30, 2005. The average active users per day is 11K (with mean availability >0.2), whose mean continuous online time is 5.7 h, and mean continuous offline time is 9.5 h. Since we do not take the repair strategy when users leave the system permanently, we randomly select 344 peers who are alive during the period for simulation and analysis.

5.2 Measurement methodology

5.2.1 *Simulation setup*

The selected 344 peers are organized as a P2P storage system. The peers connect each other randomly, assuming that connectivity probability of two peers is $m \in [0, 1]$, when $m < 1$, the P2P system is decentralized; when $m = 1$, this P2P system is centralized. The up/down behaviors of peers are driven by the real trace, reflecting the users' up/down patterns and availabilities. In the simulation, we assume that each peer can store 20 files with the same size. The ratio of total storage space to total files' size is redundancy $r^* = \sum_{i \in P} s_i / \sum_{j \in F} f_j$, which is a controllable factor in the simulation. To reduce randomness, we run every simulation 20 times, and get the mean result.

5.2.2 *Simulator design*

1) Simulator of popular file-sharing storage system. Setup the simulation environment according to subsection 5.2.1, each peer gets the pattern bit-string from the first two weeks trace, and distributes the data using random placement algorithm and "Different Levels and Pattern-complemented" placement algorithm respectively. The availabilities of files within each hour are checked by using the trace of the remaining 16 days. We utilize the following metrics: 1) the mean user-experienced availability $\overline{A}$ and its distribution, 2) the variance of user- experienced availabilities. Note that, the partition placement algorithm in [9] is a special case of "Different Levels and Pattern-complemented" placement algorithm, when the length of availability probability vector is 1, so we do not compare them separately.

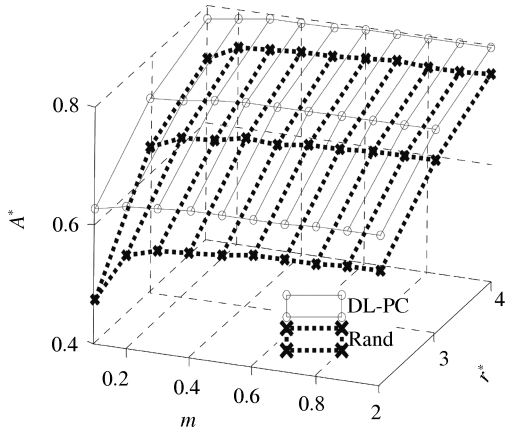

Figure 1 Mean user-experienced availability.

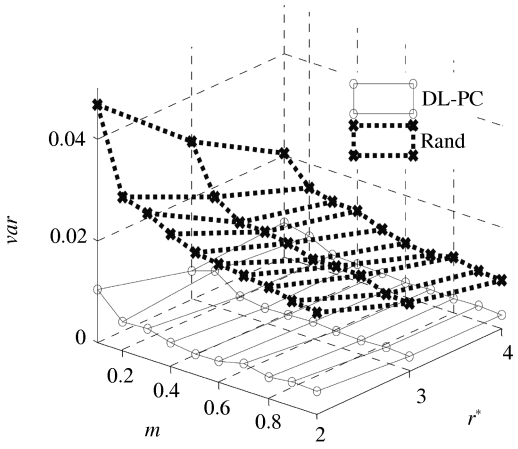

Figure 2 Variance of user-experienced availabilities.

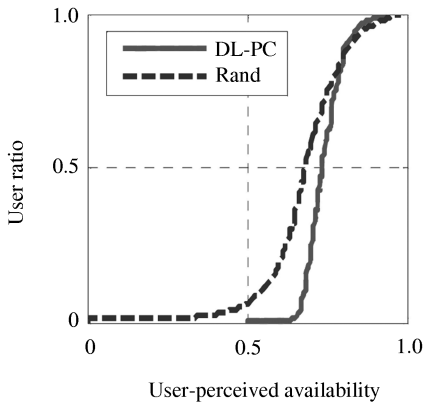

Figure 3 CDF of user-experienced availabilities ($m = 0.5, r^* = 3$).

2) Simulator of back-up storage system. Similar to the former one, but distribution strategies are random placement algorithm and "Same Level and Pattern-similar" placement algorithm. The metrics are 1) the mean user-experienced availability $\overline{A}$ and its distribution, 2) the correlation degree of peers' contribution (the availability itself) and its user experience.

5.3 Simulation results

In this section, we compare the performance of random placement algorithm and our new algorithms on enhancing the user-experience.

5.3.1 *Popular file-sharing storage system*

In the simulator of popular file-sharing storage system, we compare the performance of the random placement algorithm (Rand) and "Different Levels and Pattern-complemented" algorithm (DL-PC) under different connectivity and redundancies, the metrics is described in subsection 5.2.2. The simulation results indicate that using the new algorithm we can get higher mean user-experienced availability, and reduce the difference of users' experiences and prevent the occurrence of over-low availability.

Figure 1 shows that under different number of replicas (r^*), the changes of the mean user-experienced availability A^* of all users with the increasing of connectivity (m). We can get 1) the mean user-experienced availability is enhanced by 9% when using the new algorithm; 2) with the increasing of replicas, the performance of Rand approaches the performance of DL-PC, because it is much easier to get the effect of "Different Levels and Pattern-complemented" when there are more replicas to be selected randomly; 3) the smaller the connectivity is, the smaller the mean user-experienced availability is, espec-

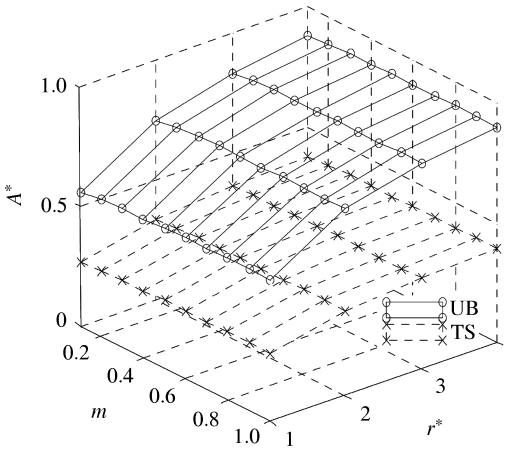

Figure 4 Availabilities evaluated by different models (using SL-PS algorithm).

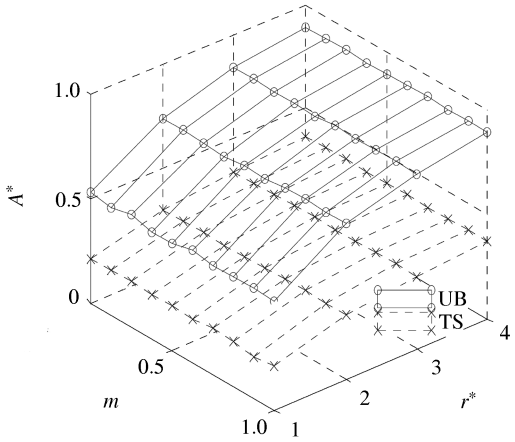

Figure 5 Availabilities evaluated by different models (using Rand algorithm).

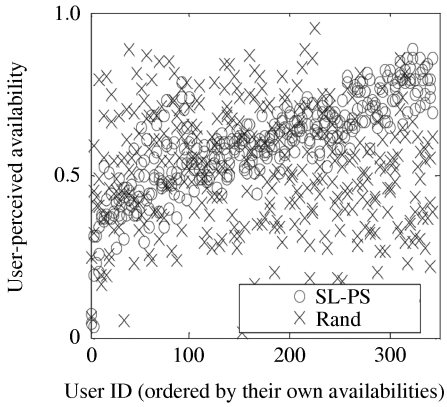

Figure 6 Distribution of user-experienced availabilities ($m = 0.5, r^* = 1$).

ially using Rand, and the mean user-experienced availability is stable when the connectivity is larger than 0.1.

Figure 2 presents the changes of the variance of user-experienced availabilities (var) with the increasing of connectivity, under different number of replicas. We can get 1) the variance when using DL-PC is much less than the variance when using Rand, and is very stable, which proves that it is much fairer to allocate resources when using our new algorithm; 2) with the increasing of replicas, the variance when using Rand is decreasing (but still higher than DL-PC).

Figure 3 shows the CDF of user-experienced availabilities under different connectivity and different replicas (due to space limitation, here we just show the situation when $m = 0.5, r^* = 3$). Apparently, the curve of DL-PC shifts right to the curve of Rand, most spots group around the high availability level; and the over-low availability spots are eliminated (there is no long-tail on the left side).

5.3.2 *Backup storage system*

In the simulator of backup storage system, we compare the performance of the random placement algorithm (Rand) and "Same Level and Pattern-similar" algorithm (SL-PS) under different connectivity and redundancy. The metrics are described in subsection 5.2.2. The results show that our new algorithm SL-PS can guarantee the mean user-experienced availability, and provide the service in proportion to users' contribution.

Figure 4 shows the availabilities evaluated by user-experience-based availability model (UB) and the traditional availability model (TS) when using SL-PS algorithm, under different replicas (r^*) and con-

nectivity (m). Figure 5 is a similar evaluation of Rand algorithm. Obviously, there is large difference (0.3–0.5) between traditional system availability and user-experience availability. SL-PS algorithm can get a little higher user-experienced availability (due to the space limitation, here we just show the situation when $m = 0.5, r^* = 1$). When using SL-PS algorithm, peers can get better experience with the increasing of their own availability. Whereas under Rand algorithm, the availabilities perceived by users are random, so each peer has the motivation of being a free-rider, which leads to disruption of the system.

6 Discussion

In order to apply our model and data placement algorithm, in this section we discuss some issues we might encounter in real systems as well as how to solve these issues by extending our algorithms.

In the "Different Levels and Pattern-complemented" algorithm, we assume the popularity of each file is similar to each other. If the popularities are not similar, we can integrate the dimension of popularity to further optimize the data placement. There have been many studies focusing on the popularity to optimize the data placement, in which the key is to assign the popular files to the peers with high availability to store. For compatibility with this dimension, we can multiply the file's availability vector with a weight (normalized to the popularity). Assuming that there are N files, the access probability of the file i is r_i, thus its availability vector can be extended to $\langle A_{i1}, A_{i2}, \ldots, A_{in}\rangle r_i / \Sigma_{i=1}^{N} r_i$. Therefore, the target of DL-PC is to maximize the weighted availabilities under the condition of different file popularity. The extended algorithm takes the popularity of files into consideration, so it can assign the popular files to the peers with high availability.

In the "Same Level and Pattern-similar" algorithm, we regard the access mode as its own up/down mode. However, due to the difference of properties between files, users may hope to have more flexible access mode. Thus, users can be allowed to modify the access vector of each file. In order to provide better service quality to the peers with high contribution, users still select peers with the similar up/down pattern and access mode in the same level.

7 Conclusions

With the purpose of setting up P2P storage system under highly dynamic environment and limited user resources, this paper provides the user-experience-based availability analysis model, integrated with users' up/down patterns, assess patterns and incentive mechanism. Based on this model, we propose decentralized data placement algorithms for two typical P2P storage applications. By the trace-driven simulation, we prove that our methods can enhance the availability perceived by users greatly, reduce the variance of the availability dramatically and eliminate the nodes with low availability in file-sharing applications; meanwhile, we can guarantee the most active users' experience, provide different-leveled service to encourage users and establish an effective motivation mechanism.

Acknowledgements

This work was supported by the National Basic Research Program of China (Grant No. 2011CB302305), the National Natural Science Foundation of China (Grant Nos. 60873051, 61073015), and the MoE-Intel Joint Research Foundation MOE-INTEL-09-06.

References

1 Bhagwan R, Tati K, Cheng Y, et al. Total recall: system support for automated availability management. In: Proceedings of the 1st ACM/Usenix Symposium on Networked Systems Design and Implementation, San Francisco, USA, 2004. 337–350

2 Bhagwan R, Savage S, Voelker G. Replication strategies for highly available peer-to-peer storage systems. In: Proceedings of FuDiCo: Future Directions in Distributed Computing, Bertinoro, Italy, 2002

3 Weatherspoon H, Kubiatowicz J. Erasure coding vs. replication: a quantitative comparison. In: Proceedings of IPTPS, Cambridge, USA, 2002
4 Blake C, Rodrigues R. High availability, scalable storage, dynamic peer networks: pick two. In: Proceedings of the 9th Workshop on Hot Topics in Operating Systems, Lihue, Hawaii, USA, 2003
5 Chun B G, Dabek F, Haeberlen A, et al. Efficient replica maintenance for distributed storage systems. In: Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation, San Jose, USA, 2006
6 Adya A, Bolosky W J, Castro M, et al. FARSITE: federated, available, and reliable storage for an incompletely trusted environment. In: Proceedings of OSDI, Boston, USA, 2002
7 Kubiatowicz J, Bindel D, Chen Y, et al. OceanStore: an architecture for global-scale persistent storage. In: Proceedings of ASPLOS, Cambridge, USA, 2000
8 Ramanathan M. Increasing object availability in peer-to-peer systems. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, 2004
9 Lin W K, Ye C, Chiu D M. Decentralized replication algorithms for improving file availability in P2P networks. In: Quality of Service, 15th IEEE International Workshop, Evanston, USA, 2007. 29–37
10 Kim K. Time-related replication for P2P storage system. In: Proceedings of the 7th International Conference on Networking, Cancun, Mexico, 2008
11 Tian J, Yang Z, Dai Y F. A data placement scheme with time-related model for P2P storages. In: Proceedings of the 7th IEEE International Conference on Peer-to-Peer Computing, Galway, Ireland, 2007
12 Liu H Y. Feature analysis of the resources and use behaviors in P2P file-sharing system Maze. Dissertation for Master's Degree. Beijing: Peking University, 2005
13 Gao Q, Yang Z, Tian J, et al. A hierarchically differential P2P storage architecture. J Software, 2007, 18: 2481–2494
14 Tian J, Dai Y. Understanding the dynamic of Peer-to-Peer systems. In: Proceedings of the 6th International Workshop on Peer-to-Peer Systems, Bellevue, USA, 2007
15 Yang M, Zhang Z, Li X, et al. An empirical study of free-riding behavior in the Maze P2P file-sharing system. In: Proceedings of IPTPS, Ithaca, USA, 2005
16 Lian Q, Peng Y, Yang M, et al. Robust incentives via multi-level Tit-for-Tat. Int J Concurr Comp, 2008, 20: 167–178