Bandwidth-Aware Replica Placement for Peer-to-Peer Storage Systems

Yu-Chih Tung
Department of Computer Science
and Information Engineering
National Taiwan University, Taiwan

Kate Ching-Ju Lin
Research Center for
Information Technology Innovation
Academia Sinica, Taiwan

Cheng-Fu Chou
Department of Computer Science
and Information Engineering
National Taiwan University, Taiwan

***Abstract*—Peer-to-Peer (P2P) storage systems are cost-effective and reliable platforms that enable users to share their storage to support variant emerging applications, such as peer-to-peer social networks and distributed backup systems. Because different users have heterogeneous online characteristics and bandwidth capabilities, how to replicate data at suitable peers has become an important issue to ensure that users can access any replica with a high probability. Previous work on data replication in a P2P storage system only considers online characteristic of each user, and aims at increasing *data availability*. However, we notice that, without considering the bandwidth capability of each user, a system might replicate popular data at a user who has a long online duration but does not have enough bandwidth capability to support all the requests. Therefore, in this work, we propose a swap-based replication scheme that jointly considers online characteristic, data popularity and bandwidth capability to improve not only *data availability*, but also *access probability* for each data item.**

I. Introduction

Peer-to-Peer (P2P) storage systems are cost-effective and reliable platforms that support variant emerging applications, such as P2P social networks (e.g., PeerSoN [1]) and distributed backup systems (e.g., Wuala [2]). In a P2P storage system, each user contributes its storage to buffer data of other peers forming a virtual storage system. Such a system potentially provides a higher *data availability*, typically referred to the probability of having at least one online replica in the system, because users could access any replica even if the original owner is off-line.

Data availability in a P2P storage system, however, is highly related to how users select peers to replicate data. Specifically, each peer has a limited storage size, and might have different online characteristics, including online time and durations. Therefore, past work on P2P storage systems [3] mainly focuses on how to choose a proper set of peers to replicate data with consideration of peers' online characteristics to maximize *data availability*. However, we notice that such bandwidth-oblivious systems that consider only online characteristic are not sufficient because an online peer might have a popular replica but can not serve all requests due to its limited bandwidth. Therefore, a more challenging issue is how to place replica to enhance the *access probability*, i.e., the probability that an access request is served successfully, by jointly considering users' online characteristics, the heterogeneity in bandwidth of each user and popularity of each data item.

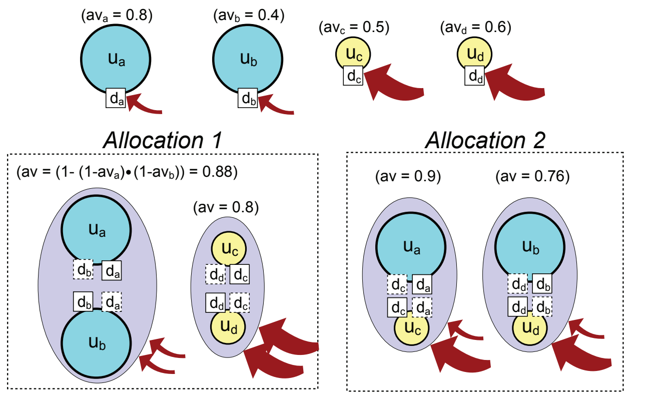


Fig. 1. Example that illustrates why bandwidth-oblivious P2P storage systems cannot provide a high *access probability*.

Consider the scenario in Fig. 1 as an example, where there are four users, u_a, u_b, u_c and u_d. Say u_a and u_b have a higher bandwidth whereas u_c and u_d have less bandwidth. Let the size of arrows in the figure represent the popularity of the indicated data item. Assume that each user can only replicate its data to one another peer, and reciprocally buffer the replica for that peer. If we only consider maximizing *data availability* for the replication group with the lowest *data availability* (called the worst group for short), the best replica placement would have u_a and u_b form the first group to replicate data of each other and make u_c and u_d form the second replication group, as *Allocation 1* shown in Fig. 1. Since both u_c and u_d buffer popular objects d_c and d_d but with a low bandwidth capability, as a result, they may not have enough bandwidth to support all the requests. Instead, a better placement strategy would let u_a and u_c form a group and let u_b and u_d form another group as *Allocation 2*. By balancing the access load among users who have a higher bandwidth, the resulting system can support a higher *access probability* for the worst group even if *data availability* of the worst group is not maximized.

Motivated by the above example, we propose *bandwidth-aware* P2P storage systems that maximize the *access probability* for the worst replication group. We first derive an analytic metric to estimate *access probability* of each data item by additionally considering user bandwidth capability and data popularity. We then prove that classifying users into replication groups that maximize the *access probability* of the worst group is an NP-complete problem, and propose a swap-based heuristic to approximate the optimal replica placement. Our evaluation results show that the proposed scheme improves the *access probability* of the worst group by 3.5 times as compared to the bandwidth-oblivious scheme.

The remainder of this paper is organized as follows. Section II contains the related work on P2P storage systems. Section III defines the bandwidth-aware replica placement problem, and then describe the proposed swap-based algorithm. Section IV evaluates the system performance. Finally, Section V concludes this paper.

II. Related Work

Data replication has been widely applied in data center networks to enhance data reliability. However, most of these literatures consider an environment where each replication machine is always connected to the system. On the contrary, P2P storage systems leverage storage resources of peers with unmanageable online behavior and access load. Therefore, the replica placement schemes designed for data center networks can not be applied to P2P storage systems directly due to network dynamics of P2P systems.

The architecture of P2P storage systems is first studied in [4]–[6]. Some coding-based schemes, such as [7], then harness erasure coding to encode a replica to smaller fragments for a higher *data availability*. In such schemes, an object needed to be recovered from system failure of certain replication machines. To further decrease the cost of recovering replica from system failure, some reactive algorithms are proposed in [8] to manage objects in P2P storage systems, while another proactive scheme [9] mitigates overhead spike based on the predefined profiles of user availability.

Replica placement algorithms are studied in [3], [10] to enhance *data availability*. However, these work only concerns user online characteristics. Our work differs from them in that it additionally considers the impact of heterogeneity in user bandwidth capability and data popularity.

III. Bandwidth-aware Replica Placement

We consider a system that consists of N users. Each user u_i provides S units of storage to the system, and wants to replicate one unit of data, d_i, at another peer. Let av_i denote availability (i.e., online probability) of user u_i, and let B_i denote the maximum number of requests that user u_i's bandwidth can support simultaneously. Furthermore, the popularity of data d_i is represented as A_i, which is estimated by the average number of access requests to d_i per time unit.

We investigate a group reciprocity-based replication system similar to the scheme in [3], where users are allocated to different groups and each user is forced to replicate the data items owned by other users in the same group. In such a group reciprocal-based replication system, user with S units of storage can thus replicate data for S other users. Therefore, $(S+1)$ users form a replication group G. Then, N users in the system can be classified into $K = \lceil N/(S+1) \rceil$ groups $G_1, G_2, \cdots, G_K$.

Consider a particular group G_k. Say that user u_i (the owner of data d_i) belongs to group G_k. Every member in G_k has a replica of d_i and therefore can be a candidate to serve all requests toward d_i. Thus, the *data availability* of d_i can be computed by $1 - \prod_{\forall u_i \in G_k}(1 - av_i)$, which equals the probability that at least one member in G_k is online.

The above availability metric is used in prior work [3] to classify users into groups such that *data availability* of the worst group can be maximized. However, even though a request to d_i can be satisfied by any online candidates, those online users might not have enough bandwidth to serve all the requests. In this situation, some of requests will be rejected if all the online candidates are fully occupied. To avoid such situations, we first derive a novel metric, called *access probability* $p(d_i)$, to compute the probability that at least one online candidate still has enough bandwidth to serve a request to object d_i. We then use this new metric to develop a swap-based heuristic (SBH) to select members for each group and improve the *access probability* for the data items cached in the worst replication group.

A. Estimating Access Probability

Say that data d_i is stored in replication group G_k. Note that the *access probability* of d_i depends on how many candidates who own the replica are online. Therefore, we define a new set $\mathcal{G}$ to represent all the online members in group G_k, and therefore $\mathcal{G} \subseteq G_k$. For example, $\mathcal{G} = \{u_1, u_2\}$ means that $u_1, u_2 \in G_k$ are online while others in G_k are offline.

Then, the number of requests to data d_i that can be supported by group G_k is determined by the aggregated bandwidth of those online members. If the aggregated bandwidth of all online members is high, then the replication group can serve more simultaneous requests. Let $\beta(\mathcal{G})$ denote the aggregated bandwidth of all online members in $\mathcal{G}$, i.e., $\beta(\mathcal{G}) = \sum_{\forall u_i \in \mathcal{G}} B_i$. Namely, those online users can serve at most $\beta(\mathcal{G})$ requests. On the other hand, let $\alpha(G_k)$ represent the aggregated data access rate of all data replicated in group G_k, which can be computed by $\alpha(G_k) = \sum_{u_i \in G_k} A_i$. In other words, $\alpha(G_k)$ equals the number of requests sent to group G_k per time unit. If $\alpha(G_k)$ is higher than $\beta(\mathcal{G})$ on average, it is very likely that a request to the replication group G_k will be dropped due to the limited bandwidth of online members.

To compute the *access probability* $p(d_i)$, we introduce an additional assumption that the average data download time (T_a) is far less than the average time a user stays after connecting to the system. We believe that this assumption is reasonable because users usually do not switch between on and off as frequently as the data access rate. Based on this assumption, the *access probability* $p(d_i)$ can be computed by

$$p(d_i) = 1 - \sum_{\forall \mathcal{G} \subseteq G_k} P(\mathcal{G}) P(\,access\, fail \,|\, \mathcal{G}), \tag{1}$$

where $P(\mathcal{G})$ is the probability that the set of online members in G_k happens to be $\mathcal{G}$, and $P(access\ fail\ |\mathcal{G})$ represents the conditional probability that, given the set of online members $\mathcal{G}$, the request cannot be served and thereby failed. Therefore,

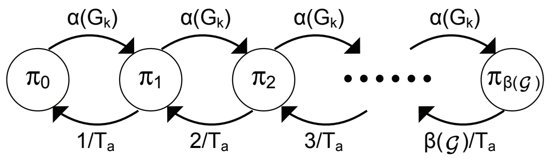


Fig. 2. Markov Chain Model for a given set of online members $\mathcal{G}$

the summation of $P(\mathcal{G})$ multiplied by $P(\,access\,fail\,|\mathcal{G})$ for all possible sets $\mathcal{G}$ represents the probability that a request to the data item d_i is failed.

Since all possible sets of online members $\mathcal{G}$ of G_k are independent to each other, the probability, $P(\mathcal{G})$, is the same as the probability that all members $u_i \in \mathcal{G}$ are online while remaining $u_j \in G_k \backslash \mathcal{G}$ are offline, and can be formalized as

$$P(\mathcal{G}) = \prod_{\forall u_i \in \mathcal{G}} av_i \prod_{\forall u_j \in G_k \backslash \mathcal{G}} (1 - av_j) \qquad (2)$$

We then use the Markov Chain Model as shown in Fig. 2 to estimate the conditional failure probability for a given set $\mathcal{G}$, i.e., $P(access\ fail\ |\mathcal{G})$. In this figure, each state π_j represents that there exist j unfinished requests in group G_k. We set the transition probability from π_j to π_{j+1} as the aggregated data access rate $\alpha(G_k)$ because all requests to any data replicated in G_k should be processed by G_k. The transition probability from π_j to π_{j-1} is set as j/T_a because we assume that every request is independent to each other and the average duration of a request is T_a, i.e., the service rate equals $1/T_a$. As a result, the conditional probability that an access request is failed given the set of online members $\mathcal{G}$ equals the probability of state $\pi_{\beta(\mathcal{G})}$, which means the probability that no more requests can be accepted by any online members in $\mathcal{G}$. This probability can be computed by the steady state equation of the $M/M/m/m$ model as follows.

$$P(\,access\,fail\,|\mathcal{G}) = \frac{(\alpha(G_k)T_a)^{\beta(\mathcal{G})}}{\beta(\mathcal{G})! \sum_{x=0}^{\beta(\mathcal{G})} \frac{(\alpha(G_k)T_a)^x}{x!}} \qquad (3)$$

By substituting the equations of $P(\mathcal{G})$ and $P(\,access\,fail\,|\mathcal{G})$ into Eq. (1), we can calculate the *access probability* $p(d_i)$.

B. Bandwidth-Aware Replication Allocation

Given the metric of *access probability* $p(d_i)$ for each data item d_i, our goal is to optimally allocate users to different replication groups such that the *access probability* $p(d_i)$ of the worst group can be maximized. However, such a group member selection problem is an NP-complete problem. Prior work [3] has reduced the 3-partition problem, which is NP-complete, to the problem of classifying users into proper replication groups such that the *data availability* of the worst group can be maximized. As a result, we can further reduce the *availability maximization problem* to our *access probability maximization problem* by assigning each user an infinite bandwidth. Given the infinite bandwidth, the *access probability* $p(d_i)$ can then be interpreted as the *data availability* of d_i because all online users must not reject any requests. Hence, if we can find any algorithm which is easier than NP-complete to solve our problem, then we can use it to solve the *availability maximization problem*. By contradiction, we get that the *access probability maximization problem* is also NP-complete.

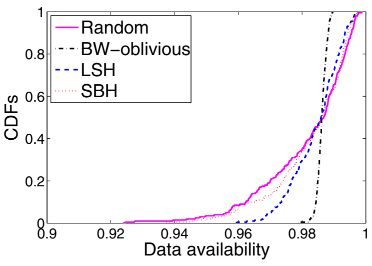
Thus, to solve such a problem with a reasonable complexity, we propose a swap-based heuristic (SBH) that classifies users into different replication groups such that all data items have a similarly high *access probability*. The design of swap-based heuristic (SBH), initially, assigns users to different replication groups randomly. Then, we start the swapping procedure that swaps two members from different groups repetitively. Specifically, in each swapping iteration, we pick two groups G_a and G_b, and swap one user u_a in G_a with another user u_b in G_b. The two issues needed to be addressed are that, first, how to select replication groups G_a and G_b to perform swapping, and, second, how to choose users u_a and u_b from them to swap.

To choose the swapping groups, we need to first define the *access probability* of a replication group G. We note that since every member in the same group stores the same set of data items, all data items replicated in a group therefore have the same *access probability*. Consequently, we can use the *access probability* of any data d_i stored in a group G to represent the *access probability* of group G (denoted by $p(G)$). Then, we perform swapping between groups G_{max} and G_{min}, where G_{max} is the group with the highest *access probability*, i.e., $G_{max} = \arg\max_G p(G)$, while $G_{min} = \arg\min_G p(G)$, to ensure that every swapping iteration enhances the worst *access probability* $p(G_{min})$ in the system.

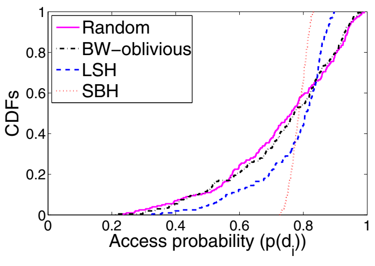
To choose which pair of users u_a and u_b to be swapped, we can exhaustedly search every possible pair of users from group G_{max} and G_{min}, and compute the new *access probability* $p'(G_{max})$ and $p'(G_{min})$ after swapping each pair of users u_a and u_b. We then select the pair that minimizes $|p'(G_{max}) - p'(G_{min})|$ and ensures $min(p'(G_{max}), p'(G_{min})) > p(G_{min})$. Choosing the best swapping pair based on this strategy promises the highest new *access probability* $p'(G)$ for the worst group after swapping.

We note that the complexity of such an exhausted search only depends on the number of members in a replication group, instead of the total number of users N. Since the number of members in a group is bounded by $S+1$ due to the limited storage size, the complexity of each swapping iteration equals O(S^2) and is a reasonable cost if S is not a significant large value. Another advantage of our proposed SBH algorithm is that it can tolerate incremental scalability of the system. That is, if a new user joins the system, we only need to perform a few more iterations of swapping for the existing replication groups, instead of reallocating all replication groups.

However, the above procedure requires the information about *access probability* for all groups and data items, and therefore might cost a high message overhead if the system scales up. Hence, we relax the above swapping algorithm to a distributed version, called the distributed swap-based heuristic (D-SBH). In D-SBH, we apply the same strategy to select users to be swapped, while choosing two swapping groups G_a and G_b at random without any global knowledge. Due to the lack of global information, the distributed algorithm D-SBH



(a) CDF of *data availability*



(b) CDF of *access probability*

Fig. 3. Results of *Data availability* and *access probability*

might require more swapping iterations to achieve a similar performance as compared to SBH.

To further balance the trade-off between performance and message overhead, we consider another variation, called hybrid SBH (H-SBH). In H-SBH, we randomly sample H replication groups in every swapping iteration, and select the group with the highest *access probability* and the group with lowest *access probability* from H sampled groups to perform swapping. In other words, if H is set to the total number of groups in the system, then H-SBH is equivalent to the original SBH scheme. On the contrary, if H is set to 2, H-SBH is then equivalent to D-SBH. Intuitively, a larger H requires a higher overhead to collect information of groups, while achieving a better performance with fewer swapping iterations.

IV. Performance Study and Discussion

We conduct discrete-event based simulations to evaluate the performance of our swap-based algorithm. Each simulation continues for 100,000 time slots. We consider a system with 1,000 users, and each user provides four units of storage to cache the data items for other peers (i.e., five members in a replication group). For each user u_i, its availability av_i follows the uniform distribution $U[0.2, 0.9]$, and its bandwidth capability B_i follows the exponential distribution with mean 200 (replica accesses from u_i per time unit). Moreover, arrival and departure of request events to data d_i follow a Poisson process with the mean arrival rate $\lambda = A_i$ and the mean departure rate $\mu = 1/T_a$. To simulate heterogeneous data popularity, we select the access rate A_i of data d_i based on the exponential distribution with mean 100 (requests per time unit). All the results reported in this paper are the average of 10 simulation runs.

We compare the following schemes: 1) Random, where users are randomly classified into different disjoint groups, 2) bandwidth-oblivious (BW-oblivious) [3], which sorts all users based on their availability and assigns the user with the maximum availability to the worst group which is not full yet, 3) linear-sorted heuristic (LSH), which is similar to BW-oblivious [3] with the difference that the users are sorted based on *access probability*, and 4) SBH, which is our proposed swap-based bandwidth-aware algorithm. We evaluate the performance in terms of the following metrics:

- Worst access probability p_{worst}: Among all data items, we examine the item that has the minimum *access probability*, and use it to represent the worst *access probability* $p_{worst} = \min_{\forall d_i} p(d_i)$ in the system.
- Fairness (p_{fair}): We compute the fairness in terms of *access probabilities* of all data items using Jain's fairness metric, i.e., $p_{fair} = \frac{(\sum_{\forall d_i} p(d_i))^2}{N \sum_{\forall d_i} p(d_i)^2}$, which ranges between $1/N$ (the most unfair case) to 1 (the fairest case).

TABLE I
Worst access probability p_{worst} and fairness p_{fair}

	Random	BW-oblivious	LSH	SBH
p_{worst}	0.2349	0.2121	0.3254	0.7264
p_{fair}	0.6881	0.6951	0.7905	0.9863

A. Performance Comparison

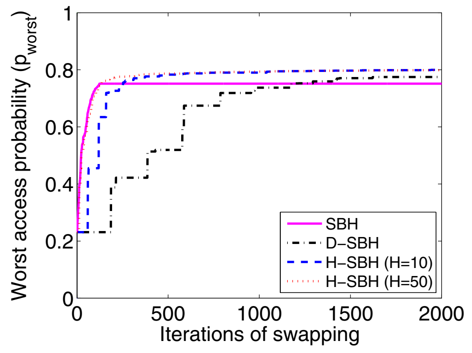
Fig. 3 plots the CDF of *data availability* and *access probability* $p(d_i)$, respectively, for all data items. This result shows that the bandwidth-oblivious scheme can provide the highest *data availability* for the worst group among all approaches if we do not care whether the online candidates have enough bandwidth to serve the requests. However, without considering bandwidth capability of each user, BW-oblivious produces an worst *access probability* (p_{worst}) as low as that in the random assignment scheme. The results reveal that even though the bandwidth-oblivious scheme provides a higher probability to find an online replica, those online replica might not be capable to be downloaded due to the congested access requests.

On the other hand, linear sorted heuristic (LSH) is a modified version of the algorithm proposed in [3], which prioritizes users by their *access probabilities*. Because the proposed *access probability* metric $p(d_i)$ jointly considers the impact of user availability, data popularity and bandwidth capability, Fig. 3 shows that LSH can therefore achieve a higher *access probability* as compared to the bandwidth-oblivious scheme. However, the improvement of LSH is not as obvious as the swapping-based algorithm, because adding a user with the highest *access probability* to the worst group may not improve its *access probability* the most. For instance, if the members in the worst group have enough bandwidth but have a low online probability, this group would favor to include a member with a high online probability rather than a high bandwidth. Therefore, although the complexity of LSH is as low as the bandwidth-oblivious algorithm, its performance is limited since *access probability* is not a linear metric.

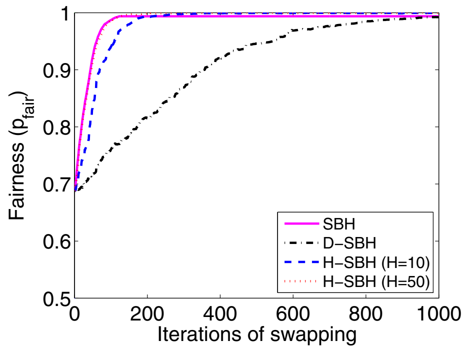
Fig. 3 also shows that SBH (with 100 swapping iterations) can provide the worst group a reasonable high *access probability* $p_{worst} = 0.7264$, as shown in Table I. It also ensures that each data item has a similar *access probability*, and, hence, achieves a fair allocation with $p_{fair} = 0.9863$, which is close to the perfect fairness (i.e., equal to 1).

B. Sensitivity Study

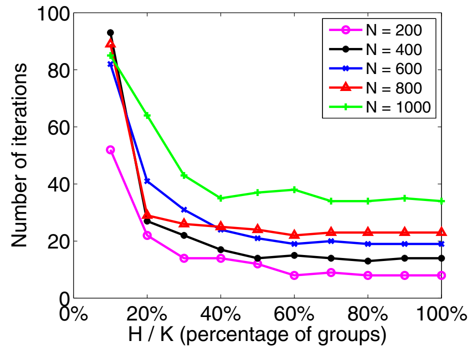
We next examine how SBH performs under various parameter settings. We note that the complexity of SBH is closely related to the number of swapping iterations. Fig. 4(a) and Fig. 4(b) show the results of the worst *access probability*



(a) Worst case p_{worst} in different SBH schemes



(b) Fairness p_{fair} in different SBH schemes



(c) Iterations needed to ensure $p_{worst} > 0.7$

Fig. 4. Performance comparison between variations of SBH schemes

p_{worst} and the fairness value p_{fair} for various swapping iterations. These figures show that SBH can converge faster than other schemes because it has the global view of all data's *access probability* $p(d_i)$ and, therefore, can select two suitable groups to perform swapping efficiently. Specifically, to provide a worst *access probability* p_{worst} higher than 0.7, SBH and H-SBH (with a coefficient $H = 50$) only need about 81 to 89 iterations to reach the threshold, while D-SBH and H-SBH (with a coefficient $H = 10$) require 787 and 160 iterations, respectively, to achieve this predefined threshold.

Even though SBH requires fewer iterations to achieve convergence, it might not be scalable for a large peer-to-peer system. In addition, Fig. 4(a) also shows that SBH converges to the worst *access probability* $p_{worst} = 0.7513$, whereas other SBH schemes can reach a higher p_{worst} with enough iterations of swapping. This is because SBH always chooses the groups with the highest and lowest *access probabilities* to perform the swapping procedure, and therefore can only achieve the local optimum. By contrast, the randomness of D-SBH and H-SBH give them an opportunity to jump out the local optimum and achieve a higher worst *access probability*.

Another interesting observation is that H-SBH (with $H = 10$) needs only a few more iterations to reach the same performance as compared to SBH, but its computational cost is almost as low as D-SBH. Furthermore, H-SBH with $H = 50$ can converge almost as fast as SBH, and requires much less computational cost than SBH in a large-scale system. Fig. 4(b) also shows that the fairness performance of all schemes follows a similar trend discussed above.

So far we have shown that the H-SBH can achieve an effective trade-off between the convergence speed and the message overheads. We next investigate what is an appropriate coefficient H to achieve the best balance between these two issue. Fig. 4(c) plots the number of swapping iterations required to converge to a worst *access probability* $p_{worst} > 0.7$ for various coefficients H. The x-axis of Fig. 4(c) denotes how many groups H are sampled from K groups in the system to perform the swapping procedure. The figure shows that the number of swapping iterations required to achieve convergence drops quickly when the sampling ratio $\frac{H}{K}$ is less than 30%. The number of swapping iterations however becomes stable when the sampling ratio is larger than 30%. We can conclude that increasing the sampling ratio does not always guarantee to improve the performance of the swapping procedure, and 30% is an appropriate sampling ratio to achieve a fast convergence and reduce more than half of the message overhead.

V. Conclusion

This paper investigates the bandwidth-aware replica placement problem in a P2P storage system. We model a novel metric *access probability* that jointly considers users' online characteristic, bandwidth capability and data popularity. We prove that optimizing *access probability* for the worst group is an NP-complete problem, and propose a swap-based heuristic algorithm, called SBH, to classify users into replication groups in order to approximate the maximal worst *access probability*. Our simulation results indicate that SBH can improve the worst *access probability* by about 3.5 times as compared to the bandwidth-oblivious scheme. The results also validate that the hybrid scheme H-SBH can take a balance between converging speed and the message overhead, earning system scalability.

References

[1] "PeerSoN," http://www.peerson.net/.
[2] "Wuala," http://www.wuala.com/.
[3] K. Rzadca, A. Datta, and S. Buchegger, "Replica placement in p2p storage: Complexity and game theoretic analyses," in *IEEE ICDCS*, 2010.
[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," *SIGOPS Oper. Syst. Rev.*, vol. 35, pp. 202–215, 2001.
[5] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: federated, available, and reliable storage for an incompletely trusted environment," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 1–14, 2002.
[6] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "Oceanstore: an architecture for global-scale persistent storage," *SIGARCH Comput. Archit. News*, vol. 28, pp. 190–201, 2000.
[7] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. of IPTPS*, 2002.
[8] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: system support for automated availability management," in *USENIX NSDI*, 2004,.
[9] A. Duminuco, E. Biersack, and T. En-Najjary, "Proactive replication in distributed storage systems using machine availability estimation," in *ACM CoNEXT 2007*.
[10] S. Bernard and F. Le Fessant, "Optimizing peer-to-peer backup using lifetime estimations," in *EDBT/ICDT Workshops*, 2009.