# Awake: decentralized and availability aware replication for P2P cloud storage

Yahya Hassanzadeh-Nazarabadi, Alptekin Küpçü and Öznur Özkasap
Department of Computer Engineering, Koç University, İstanbul, Turkey
{yhassanzadeh13, akupcu, oozkasap}@ku.edu.tr

***Abstract*—The traditional decentralized availability-based replication algorithms suffer from high dependence on the underlying system's churn behavior, randomness in replica selection, and the inability of maximizing the replicas availability. These drawbacks result in poor data availability especially in low available systems as well as where the churn behavior is mispredicted. In this paper, we propose dynamic, fully decentralized availability aware algorithm named Awake, with the goal of maximizing the availability of replicas. Compared to the existing solutions, Awake always provides the maximum availability of replicas regardless of the underlying system's churn behavior. By employing Awake, a data owner can select its replicas only based on the aggregated availability information of nodes obtained in a fully decentralized manner with asymptotically the same message overhead as the communication complexity of the underlying system. Awake has linear space complexity in the number of registered users to the system. Our extensive simulation results show that in comparison to the best existing decentralized solutions, regardless of the underlying churn model of the system, Awake improves the availability of replicas with a gain of about 21%. Likewise, Awake is scalable by showing the same performance independent of the system size. Employing Awake in a system with 1 million nodes has the expected space consumption of 25 megabytes on each node as well as the expected communication overhead of 480 kilobytes per message.**

***Index Terms*—P2P cloud storage, replication, availability awareness, churn.**

## I. Introduction

Peer-to-peer (P2P) cloud storages consist of a set of nodes where a node can be both a data owner and a data requester simultaneously. Data owner possesses a set of data objects and aims to share them with a group of authorized nodes called data requester nodes. In P2P cloud storages nodes oscillate between online and offline periods. A node arrives at the system and starts an online state. After a while, it terminates its session and departs from the system or fails. A departed or failed node may come back at another time to the system or may leave the system forever. This arrival and departure of the nodes to and from the P2P system are called churn [1]. Churn negatively affects the data availability in a P2P cloud storage. When a data owner goes offline or fails, its data objects would no longer be available to the data requesters.

To remedy the data availability problem of P2P cloud storages under churn, the data owner makes copies of its data objects on the other nodes of the system. Those nodes are called replicas and the operation of determining replicas is called replication [2], [3]. After replication is done, in case data owner becomes unavailable, the data requesters can utilize the replicas.

The traditional decentralized availability-based replication algorithms aim at improving the performance of the P2P cloud storage by either immediately resolving replica's failure or by providing an average number of available replicas for a long period of time e.g., providing an average number of three available replicas for one month. Probing the replicas and determining a new replica for each failure [4]–[8], randomized replication [9], cluster-based replication [10]–[13], and correlation-based replication [14] are the most common decentralized availability-based replication methods.

All of the decentralized availability-based replication algorithms employ some kind of randomness for replica selection. Employing randomness prevents the algorithms to purely consider the availability patterns of nodes in the replica selection procedure. This leads the system to cases where only a few number of replicas or even no replica is available. Poor availability of replicas results in performance degradation of the P2P cloud storage in the terms of data availability. Likewise, all of the decentralized availability-based existing solutions make explicit assumptions about the underlying churn behavior of the system for replica selection. This causes their performance to be extremely narrowed by the underlying churn behavior of the system. In the case where the churn behavior of the system is mispredicted or changes over the time, employing the existing solutions results in poor data availability.

To improve the data availability of P2P cloud storages under churn, **we propose a dynamic, fully decentralized, churn behavior independent, and availability aware replication algorithm named Awake**. The availability aware replication is defined as dividing a fixed size periodic time interval (*FPTI*) into a set of identical time slots (*TS*) and placing the replicas such that the maximum availability of replicas during each *TS* is achieved. By employing Awake, a data owner can replicate its data objects in a fully decentralized manner with the availability awareness of the replication achieved. For instance, considering *FPTI* as a day and *TS* as an hour, compared to the best existing solutions our approach provides maximum availability of replicas during each hour of the day.

Awake can be employed in any structured P2P cloud storage given that the availability information of nodes is piggybacked on the messages that they route or initiate. The size of message overhead is asymptotically identical to the communication

complexity of the underlying structured P2P system. For instance, employing Awake in a Distributed Hash Table (DHT)-based [4] cloud storage with the capacity of $N$ nodes that have the communication complexity of $O(\log N)$, costs the message overhead of size $O(\log N)$.

The contributions of this study are:

- We propose Awake: **the first churn behavior independent, dynamic, fully decentralized availability aware** replication algorithm for P2P cloud storages.
- Space complexity of Awake is linear in the number of registered users to the system.
- Communication overhead of Awake is the same as the communication complexity of the underlying P2P cloud storage that Awake is applied on.
- We extended the SkipSim [15], [16], for simulating and evaluating the availability-based replication algorithms.
- We modeled the best known decentralized availability-based replication algorithms on SkipSim and compared with Awake.
- The simulation results show that on average regardless of underlying churn behavior of the system, Awake improves the availability of replicas with the gain of about **21%** in comparison to the best known decentralized existing solutions.
- Awake is scalable in the sense that it shows the same performance independent of the system size.
- Employing Awake in a system with 1 million nodes, charges a space consumption of **25 megabytes** on each node and a communication overhead of **480 kilobytes** on each message.

The rest of the paper is organized as follows. Section II presents the system characteristics that Awake can be applied on. In Section III, our approach Awake is described in detail. The related works and simulations setup are presented in Sections IV and V, respectively. Performance results are presented in Section VI, followed by conclusion in Section VII.

## II. Preliminaries

### A. System Architecture

Awake works on top of a structured P2P system such as a DHT. In a structured P2P system, the overlay network is based on a specific topology. In our view of a structured P2P system, nodes just use the overlay network to efficiently search for other nodes and data.

In a search operation, the node that initiates the operation is named the initiator. The node or data object that an initiator searches for is called search target. As a result of a search operation, the address of the target node or address of the node that owns the target data object is returned to the search initiator. Once the search initiator receives the address of the search target, it communicates the search target directly.

We consider a structured P2P cloud storage where each node corresponds to a unique peer in the real world. There exist two roles for each peer: data owner and data requester. Data owner owns a set of data objects and data requester is the authorized consumer of those objects. A peer can be both a data owner as well as a data requester for other data owners.

### B. Availability Information

*1) Availability Vector:* Availability pattern of a node is represented by a vector of size *S* and is called the availability vector. *S* is defined as the number of time slots and is equal to $\frac{FPTI}{TS}$. Availability vector of a node $i$ is denoted by $AV_i[]$. $AV_i[t]$ represents the availability probability of the node $i$ in time slot $t$ of the *FPTI*. Node $i$ computes its $AV_i[t]$ by dividing the total duration of its availability during time slot $t$ over the number of times that *FPTI* has repeated up to the time of computation. For example, consider a system with *FPTI* equal to a day and *TS* equal to an hour which five days have elapsed since the birth time of the system. In such a system, *S* is equal to 24 and hence the availability vector of nodes is a vector of size 24. $AV_i[t]$ is computed as the total duration that node $i$ was available at hour $t$, over five days. For example, if during the last five days the total availability duration of node $i$ during hour $t$ was 2.3 hours, $AV_i[t] = \frac{2.3}{5} = 0.46$.

*2) Availability Table:* The availability table of a node is the local availability view of it for the system. Availability table of node $i$ is represented as $AT_i[][]$ which is a two dimensional array of size $N \times S$. $AT_i[j][t]$ is the knowledge of the node $i$ about the availability probability of the node $j$ in time slot $t$ of *FPTI*.

A node piggybacks its $id$ and availability vector on the messages' header that it routes or initiates. Upon receiving a message to route or read, the receiver obtains the piggybacked availability vectors and updates its availability table accordingly. Nodes are assumed to be honest in computing and piggybacking their availability vectors as well as their *id*s. To emphasize the recent availability behavior of node $j$ while considering its availability history, the node $i$ updates $AT_i[j][t]$ using an exponential moving average mechanism.

$$\begin{aligned} &\forall t \quad 0 \leq t \leq S \\ &AT_i[j][t] = (1-\beta) \times AT_i[j][t] + \beta \times AV_j[t] \end{aligned} \tag{1}$$

Equation 1 presents update procedure of node $i$'s availability table upon receiving a message that contains the availability vector of node $j$. In this equation, the right side and left side $AT_i[j][t]$ values are the $(j,t)$ entry of node $i$'s availability table before and after the update, respectively. $AV_j[]$ is the availability vector of node $j$ and $0 \leq \beta \leq 1$ is the effect factor of amplifying the new availability probabilities over the old ones in learning the value of $AT_i[j][t]$. In this text, we call $\beta$ as the **learning factor**.

### C. Churn Model

In a P2P cloud storage, a node oscillates between online and offline states. In each online period, the node is available for a certain while named the session length. Availability of a node corresponds to its session length. As the session length elapses, the node goes offline for a certain while that is called

| Churn Model | Distribution | SL $\lambda$ | SL $k$ | Average SL (Hour) | DT $\lambda$ | DT $k$ | Average DT(Hour) | Availability Ratio |
|---|---|---|---|---|---|---|---|---|
| High Available [17] | Weibull | 0.35 | 0.34 | 1.96 | 0.179 | 0.34 | 0.99 | 1.97 |
| Moderate Available [17] | Weibull | 0.69 | 0.59 | 1.06 | 0.65 | 0.34 | 0.99 | 1.07 |
| Low Available [18] | Exponential | 0.118 | — | 8.42 | 0.017 | — | 58.32 | 0.14 |

TABLE I: Characteristics of the BitTorrent-based churn models used. SL and DT correspond to the session length and downtime distributions, respectively.

the downtime. After the downtime of the node is passed, the node comes back to the system and this flow cycles.

We assume that the connectivity of the structured P2P routing infrastructure is churn resilient. The departure or failure of a node results in connectivity disruption in the paths routed from that node. This negatively affects the connectivity of the whole system and yields failure in routing the transactions. To overcome this problem, it is assumed that a node maintains and frequently updates an alternative direct communication link between its consecutive predecessors and successors. When the node departs or is failed, its consecutive predecessors and successors use the alternative direct communication to route a query. Hence, although the churn negatively affects the data objects availability of the system, it does not affect the system connectivity.

For a churn model, we define the **availability ratio** as the ratio of its session length over its downtime. Based on the availability ratio, a churn model is high available if its availability ratio is higher than one. A churn model is moderate available if its availability ratio is around one. And a churn model is low available if its availability ratio is less than one.

Session length and downtime distributions of the churn model define the availability pattern of nodes. There are several studies on the churn models, each considers a certain system e.g. BitTorrent [17]–[20], Kad [17], Gnutella [17], [21], eDonkey [14], Kademlia [22], and PlanetLab [7], [23].

Among the churn models we reviewed, the BitTorent-based models are concrete, parametrically clear and consistent with each other, and hence were selected to model our system. Table I summarizes these churn models with their characteristics. Where SL and DT correspond to the session length and downtime distributions, respectively. These models either follow a Weibull or an Exponential distribution.

*1) Weibull-based Churn Model [17]:* In a Weibull-based churn model, the session length and downtime of nodes are modeled with Weibull distributions. A Weibull distribution is defined with two parameters: shape and scale. The shape parameter affects the shape of the distribution and is denoted by $k$. The scale parameter is denoted by $\lambda$. A large scale parameter results in the distribution to be less spread out and more concentrated around its expected value. Considering the random variable $T$ as a representation of the session length (or downtime), Equation 2 represents the Cumulative Distribution Function (CDF) of the session length (or downtime) of a system that follows the Weibull distribution. Also, Equations 3 and 4 represent the mean and variance of the corresponding Weibull distribution, respectively. In those equations, $\Gamma$ is the Gamma function. For a positive integer $n$ greater than one, $\Gamma(n)$ is defined as $(n-1)!$.

$$\Pr(T \leq t) = 1 - \exp(-(\frac{t}{\lambda})^k) \tag{2}$$

$$E[T] = \lambda \times \Gamma(1 + \frac{1}{k}) \tag{3}$$

$$var(T) = \lambda^2 \times \left[\Gamma(1 + \frac{2}{k}) - \Gamma^2(1 + \frac{1}{k})\right] \tag{4}$$

*2) Exponential-based Churn Model [18]:* In an exponential-based churn model, the session length and downtime of nodes follow exponential distributions. An exponential distribution is specified with a rate parameter ($\lambda$). The rate parameter of the downtime and session length distributions shows how often a node arrives at the system or departs from the system, respectively. Considering random variable $T$ as the session length (or downtime) of the nodes, Equation 5 represents the CDF of the session length (or downtime) of nodes in a system that follows the exponential distribution. Likewise, the mean and variance of the random variable $T$ are shown in the Equations 6 and 7 respectively.

$$\Pr(T \leq t) = 1 - exp(-\lambda t) \tag{5}$$

$$E[T] = \frac{1}{\lambda} \tag{6}$$

$$var(T) = \frac{1}{\lambda^2} \tag{7}$$

## III. Awake: Availability-Aware Replication

### A. Scenario

As a data owner joins the system, it starts searching for other nodes as well as routing other nodes' search queries. The data owner sends its availability vector to other nodes by means of piggybacking on the query messages it initiates or routes. Likewise, the data owner obtains other nodes piggybacked availability vectors from the messages it routes, and updates its availability table accordingly. This is a **learning phase** as the data owner learns about the availability behavior of the system. The learning phase is common among all the existing availability-based solutions. A large learning phase helps the data owner to obtain a more accurate availability view from the system. The data owner continues to this learning up to a point which is called the replication time.

At the **replication time**, the data owner runs Awake by giving its availability table as well as the authorized number of replicas. As Awake terminates, it outsources the replica set to the data owner and data owner replicates on the replica set.

### B. Motivations and Challenges

In Awake, the main challenge is to move toward maximizing the replica availability in a decentralized manner regardless of

the underlying churn behavior of the system. The main disadvantages of the existing decentralized solutions are considering explicit assumptions about the underlying churn behavior of the system (and hence strong dependency on it) as well as employing randomness in replica selection. The hard parts in designing Awake were to achieve a formal formulation of availability aware replication that is refined from randomized decisions and any assumption about the underlying churn behavior of the system.

Awake is the first decentralized availability aware replication algorithm with mathematical objective, formulation, and constraints. This formulation is completely free of including any assumption about the underlying churn model of the system. Heart of Awake is its ILP model of availability aware replication which its correctness is self-explanatory based on the nature of ILP models. Due to its ILP maximization objective that is clearly bounded by the constraints, being free of randomness, and being independent of underlying system's churn behavior, Awake is theoretically expected to outperform the existing solutions. This claim is supported by the empirical results.

### C. *Algorithm Overview*

Awake is a dynamic fully decentralized availability aware replication algorithm for the P2P cloud storages. By employing Awake, a data owner can determine its replicas without the need for communicating with any special node as the coordinator. Since Awake is an availability aware algorithm, by employing it in a system, maximum replication availability during each time slot is obtained. Maximum replication availability corresponds to the maximum average number of replicas which are available during each time slot. Replicating data objects of a node in Awake does not necessarily mean that the data objects are publicly available. Access control is a separate problem that can be solved, for example, by means of encrypting the data objects and delivering the keys to the authorized parties. The authorized parties are called the data requester nodes.

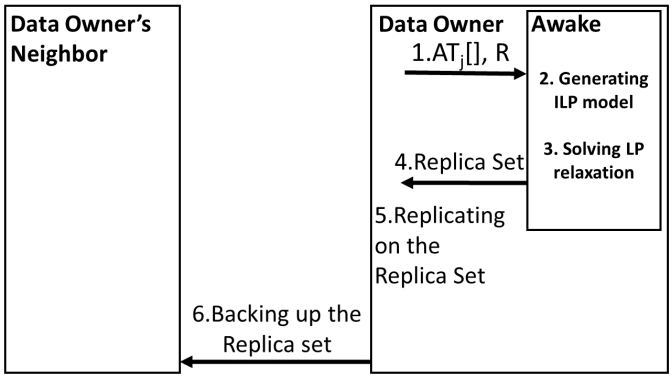


Fig. 1: The interactions between the data owner, data owner's neighbor and Awake.

Figure 1 illustrates the interactions between the data owner, data owner's neighbor, and Awake. As the input, Awake receives availability table of the data owner as well as authorized number of replicas (Step 1). The authorized number of replicas is called the **replication degree** and is represented by $R$. Awake then models the availability aware replication as an integer linear programming (ILP) (Step 2), solves the LP relaxation of it (Step 3) and provides the replica set to the data owner (Step 4). Data owner replicates its data objects on the replica set provided by Awake (Step 5).

**Data Replication vs. Replica Set Replication:** The data owner does the data replication based on the replica set that is provided by Awake. However, in order to prevent the unavailability of the replica set by failure or departure of the data owner, similar to [24], the data owner makes a backup of its replica set accompanied by an expiration time on its neighbors in the underlying structured P2P (Step 6). This procedure is called replica set replication. The expiration time is defined as the time of next execution of Awake based on the updated availability information of nodes. Likewise, the data owner's neighbors are the nodes which are directly connected to the data owner in the structured P2P routing infrastructure. The data owner is assumed to check its neighbors list frequently and back up its replica set on the new neighbors. It worth to note that Awake only provides the replica set for data replication. While the replica set replication is done independently of Awake as a fault tolerance approach.

When an intermediate node receives a search query requesting the replica set of a certain data owner, it checks whether it possesses a valid replica set backup of that data owner. A valid backup is the one that is not expired. If the intermediate node has a valid backup, it responds to the search query on behalf of the data owner by sending the replica set to the data requester node. Otherwise, the intermediate node routes the search query to the data owner via the next node.

### D. *Algorithm Description*

*1) **Input and Output**:* Considering data owner $j$ running Awake, inputs of Awake are availability table of the data owner, $AT_j[][]$, as well as the replication degree of the data owner, $R$. The output of Awake is a binary vector $Y$ with the size $N$ where $N$ is the capacity of the system in terms of nodes and is considered as a constant regardless of system arrivals and departures. The capacity of system is defined as the maximum number of users that can register to that system. Although $N$ is constant, new nodes can join the system, and old nodes may depart permanently. However, the total number of user registration to the system is assumed to be constant $N$. If node $i$ is selected as a replica, $Y_i = 1$, otherwise $Y_i = 0$.

*2) **Generating and Solving the ILP model**:* Awake models the availability aware replication as an ILP represented by Equations 8-10. The only variable of this ILP is $Y$ and constants are $N$, $R$, and $AT[][]$.

$$\max \sum_{t=1}^{S} \sum_{i=1}^{N} Y_i \times AT_j[i][t] \quad \text{s.t.} \tag{8}$$

$$\sum_{i=1}^{N} Y_i = R \tag{9}$$

$$\forall i, 1 \leq i \leq N \quad Y_i \in \{0, 1\} \tag{10}$$

*a) The objective function (Equation 8):* The objective is to maximize the total availability of replicas over all the time slots. Availability of replicas in a time slot is defined as the summation of their availability probabilities from the availability table of the data owner.

As represented by this equation, if node $i$ is selected as a replica ($Y_i = 1$) it contributes to the availability of time slot $t$ with its availability probability in that time slot ($AT_j[i][t]$). The summation over $N$ represents the total availability of replicas during time slot $t$ of *FPTI*. Subsequently, the summation over *S* represents the total availability of replicas over all the time slots.

As the total availability of replicas during each time slot is non-negative, maximizing the total availability of replicas over all the time slots implicitly results in the maximum availability during each time slot as well.

*b) The replication degree constraint (Equation 9):* If node $i$ is selected as a replica, $Y_i$ is set to one, otherwise zero. Summing up the elements of $Y$ vector hence results in the replication degree. Equation 9 shows the constraint on the replication degree.

*c) The authorized values for output variable (Equation 10):* This constraint represents that each element of the $Y$ vector should be either one or zero which shows whether the corresponding node is selected as a replica or not, respectively.

After Awake models the availability aware replication as an ILP, it solves the LP relaxation of ILP and outsources the $Y$ vector to the data owner.

## IV. Related Works

### A. Reactive Replication

Reactive replication algorithms replicate regardless of availability patterns of the nodes. They react to a replica failure by placing a new replica. In this class of algorithms, after an initial replication is done, the replicas are periodically probed, for example, every 12 hours. If a replica does not answer the probe message in a certain while, it is presumed to be failed. Failed replicas are substituted by newly placed ones [8]. Probing of replicas is done by the data owner, data requesters or other replicas.

For a large number of replicas or small probing periods, probing all the replicas is inefficient. To resolve this problem while preserving the fairness, at each probing time a subset of replicas is chosen uniformly at random and probed [4]–[7].

### B. Proactive Replication

The goal of a proactive replication algorithm is to provide an average availability of replicas for a long period of time. For example, a proactive replication algorithm may provide an average availability of 2 replicas at each hour for 3 months. A proactive replication algorithm is supposed to be executed periodically to refresh the replica selection based on the updated availability pattern of nodes up to that period. In the previous example, the proactive replication algorithm is expected to be executed every 3 months. Likewise, when the availability behavior of replicas changes negatively and degrades the system performance, the proactive replication is supposed to be executed based on the new availability behavior of nodes. There are several proactive replication algorithms that depend on a specific churn behavior of the system:

*1) Randomized Replication [9], [26]:* In a randomized replication algorithm, a number of nodes denoted by the replication degree are selected as replicas uniformly at random. Randomized replications perform well in the highly available systems where the average availability of nodes is high. Choosing replicas uniformly at random hence results in an acceptable expected number of available replicas.

*2) Cluster-Based Replication [10]–[14]:* A cluster-based replication algorithm divides the nodes based on their common features such as time zone, load, and query rate into a set of cliques. The algorithm then distributes the replication degree among the cliques considering their availability pattern. Cluster-based replication algorithms perform well in high and moderate available systems.

*3) Correlation-Based Replication [14], [25]:* Correlation is defined as the similarity between the availability patterns of the nodes. The more two nodes are correlated with each other, the more similarity they have in their availability behavior. In other words, for a pair of highly correlated nodes, whenever one node is available, with a high probability the other node would be available as well. Similarly, two nodes that have reverse availability patterns with respect to each other, are called anti-correlated nodes. For a pair of anti-correlated nodes, when one node is unavailable, with a high probability, the other one is available. The correlation between two nodes is defined as the dot product of their availability vectors. The higher the dot product is, the higher two nodes are correlated with each other. Likewise, a zero dot product shows a pair of anti-correlated nodes.

The goal of a correlation-based replication algorithm is to provide $k$-availability of replicas in the system. $k$-availability is defined as guaranteeing the availability of at least $k$ replicas at any time in the system. To achieve this goal, a correlation-based replication algorithm replicates on $k$ pairs of anti-correlated nodes. In this way, totally $2 \times k$ replicas are selected.

Considering a pair of anti-correlated replicas, when one replica is offline, the other one is available with a very high probability. For each pair of anti-correlated replicas, hence, always one replica is available. Having $k$ pairs of anti-correlated replicas results in the availability of at least $k$ replicas at any time.

| Strategy | Behavior | Availability Aware | Churn Model Independent | Objective |
|---|---|---|---|---|
| Probing [4]–[8] | Reactive | No | No | Resolving Failures |
| Randomized [9] | Proactive | No | No | Average Availability |
| Cluster-Based [10]–[13] | Proactive | No | No | Average Availability |
| Correlation-Based [25] | Proactive | No | No | Average Availability |
| **Awake** | **Proactive** | **Yes** | **Yes** | **Maximum Availability** |

TABLE II: Comparison of various availability-based replication strategies

However, finding $k$ pairs of anti-correlated nodes may not be always feasible. Thus, empirically, a correlation-based replication algorithm selects $k$ pairs of nodes with the minimum correlation as a pair of replicas. For this reason, the correlation-based replication algorithms are merely suitable for the low available systems which experience a very low pairwise correlation of nodes.

### C. Comparison

Table II shows a comparison between various availability-based replication strategies. As shown in this table, among the availability-based replication algorithms, Awake is the only availability aware one. Also, while the main objective of all the proactive solutions is to provide an average availability of replicas, Awake provides the maximum replication availability. Compared to the existing solutions, Awake is the only algorithm that works independently of the underlying system's churn model. All of the replication algorithms listed in the table are decentralized. Likewise, all of them are dynamic in the sense that a data owner can replicate at any arbitrary time.

### D. Algorithms used for comparison

Followings are the implementation details of the algorithms that are selected for the sake of comparison with Awake.

*1) Randomized Replication:* The data owner pings nodes repeatedly at random until it finds as many as available nodes as the replication degree and replicates on them [9].

*2) Cluster-Based Replication:* The data owner performs a $k$-mean clustering [27] of the nodes based on the availability of the nodes which is the second norm of their availability vector where $k$ is equal to the replication degree. After the clustering is done, the replication degree is distributed among the clusters based on their availability. The availability of each cluster is defined as the second norm of the average availability vectors of nodes inside that cluster. Based on the assigned replication degree to each cluster, the replication is done on the most available nodes of that cluster [10]–[13].

*3) Correlation-Based Replication:* Considering the replication degree as $d$, the data owner aims to find $\frac{d}{2}$ pairs of nodes with the minimum correlation.

Considering just the pairs with minimum correlation may fall the replication into a pitfall. Two nodes that are not available at all and hence have an availability vector of all zero have the minimum correlation respect to each other. However, they are the worst candidates for replication.

To avoid falling into this pitfall, $\frac{d}{2}$ pairs of replicas are initialized with the $\frac{d}{2}$ most available nodes. After this initialization, each pair has exactly one node. Let's call that node the premier of the pair. Next, to complete each pair, the most available node that has the least correlation with the premier of that pair is selected.

## V. Simulation Setup

To simulate and evaluate the decentralized availability-based replication algorithms we extended SkipSim [15], [16] by enabling it to simulate the availability-based replication algorithms. Table III represents a comparison between the versions of SkipSim before and after our extensions. The previous version of SkipSim was capable of generating static simulations i.e, the simulations where the topology is fixed and does not change. As the whole SkipSim architecture was designed in a static oriented manner, adding time feature to the simulations and making them dynamic was the most challenging part.

We embedded the high, moderate and low available churn models into the SkipSim. In the new version of SkipSim, nodes arrive and depart based on the selected churn model. We also implemented Awake as well as the state-of-the-art availability-based replication algorithms in the SkipSim. This package of the algorithms is called the dynamic replication package. By the extensions, SkipSim is able to evaluate the performance of availability-based replication algorithms from the scalability, average number of available replicas, replication time and learning factor points of view.

| Feature | Before extensions | After extensions |
|---|---|---|
| Static Simulations | ✓ | ✓ |
| Identifier Assignments | ✓ | ✓ |
| Identifier Assignment Evaluation | ✓ | ✓ |
| Static Replication | ✓ | ✓ |
| Static Replication Evaluation | ✓ | ✓ |
| Churn Models | ✗ | ✓ |
| Dynamic Simulations | ✗ | ✓ |
| Dynamic Replication | ✗ | ✓ |
| Dynamic Replication Evaluation | ✗ | ✓ |

TABLE III: A comparison of SkipSim features before and after our extensions.

For each simulation setup, we generated 100 random topologies. *FPTI* and *TS* were set to a day and an hour, respectively. Each topology was simulated for a lifetime of 3 months. A simulation step corresponds to an hour. At $t = 0$ all the nodes are available in the system. As the time proceeds, nodes depart from or come back to the system based on the session length and downtime distributions of the system's churn model. In each simulation setup, the replication algorithm is executed at the end of the second day.

For each topology, at each hour, a number of search transactions are initiated between the nodes. As shown in Equation 11, number of transactions at hour $t$ represented by $n_{trans_t}$ is computed as the binomial coefficient of number of available nodes at hour $t$ (represented by $n_{available_t}$) and 2. For a search

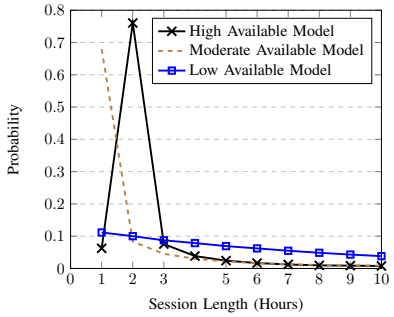


Fig. 2: Extracted churn model session lengths probability distributions from SkipSim

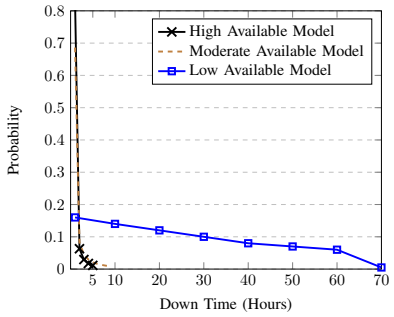


Fig. 3: Extracted churn model down times probability distributions from SkipSim

transaction done at hour $t$, the search initiator and search target are selected from the set of available nodes at hour $t$ and set of all the registered nodes to the system, respectively. For each transaction, availability table of the nodes on the transaction path including the source and destination are updated by the aggregated piggybacked availability vectors. At the replication time, a data owner is selected uniformly at random that executes the replication algorithm.

$$n_{trans_t} = \binom{n_{available_t}}{2} \tag{11}$$

We simulated each algorithm with a system size of 128 nodes under the churn models presented in Table I. Also, the scalability of each algorithm was evaluated under the moderate available churn model with system sizes of 128, 256, 512 and 1024 nodes. Additionally, following the moderate available churn model, running time, space consumption, and communication overhead of Awake were evaluated with the system sizes of 128, 256, 512, 1024, 2048, 4096, and 8192 nodes.

**Correctness of Churn Modeling:** To verify the correctness of our implementations Figures 2 and 3 show the extracted probability distributions of the session lengths and down times of churn models from SkipSim, respectively. Table IV represents the corresponding average values of Figures 2 and 3. Comparing the empirical average values (Table IV) with the theoretical average values (Table I) results in an average churn model implementation error of about 3%.

| Churn Model | Average SL (Hour) | Average DT (Hour) |
|---|---|---|
| High Available | 2.03 | 1.07 |
| Moderate Available | 1.11 | 1.01 |
| Low Available | 8.37 | 58.41 |

TABLE IV: Extracted average session length and downtime values from the SkipSim. SL and DT correspond to the session length and downtime distributions, respectively.

## VI. Performance Results

### A. Learning Factor Effect

Figure 4 shows the effect of the learning factor, $\beta$, that was presented in Equation 1, on the average available number of replicas. In this experiment, the replication degree was fixed to 4 and the algorithms were simulated under the moderate available churn model. As shown in this figure, as $\beta$ varies from 0.1 to 0.9, Awake outperforms the cluster based replication as the best existing solution with the gain of about **20%**. Since all of the replication algorithms work at their best with $\beta = 0.5$, in all simulations the learning factor was set to 0.5.

### B. Average Availability of Replicas

Figures 5.a and 5.b depict the average availability of replication algorithms under the high and moderate available churn models. Compared to the cluster-based replication algorithm as the best existing solution under these two models, Awake improves the average availability of replicas about **24%** and **14%** under the high and moderate available churn models, respectively.

In the high available churn model, the correlation between nodes is high. Therefore, correlations in the pairs of nodes that are selected by the correlation-based replication algorithm are too far from zero. This causes the correlation-based replication to perform the worst algorithm under the high available churn model. This problem is ameliorated in the moderate available churn model as the correlation between the nodes degrades, and the correlation-based replication beats the randomized replication.

Figures 5.c shows the average availability of replication algorithms under the low available churn model. Under this model, the correlation between nodes reaches its minimum and many suitable replication candidate pairs with close to zero correlation emerge. In this case, the correlation-based replication algorithm conquers the cluster-based replication and becomes the best among the existing solutions. However, under the low available churn model, Awake outperforms the correlation-based replication with the gain of about **26%**.

Considering the low available churn model, as the replication degree increases, cluster-based replication assigns replication quota to the low available cliques with higher probability. The replicas from low available cliques do not play a significant role in the performance of the cluster-based replication algorithm. This causes the performance of the cluster-based replication to reach a steady state and even is beaten by the randomized replication when the replication degree goes beyond 8 replicas as illustrated in Figure 5.c. This trend

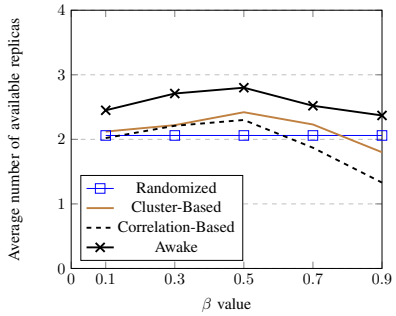
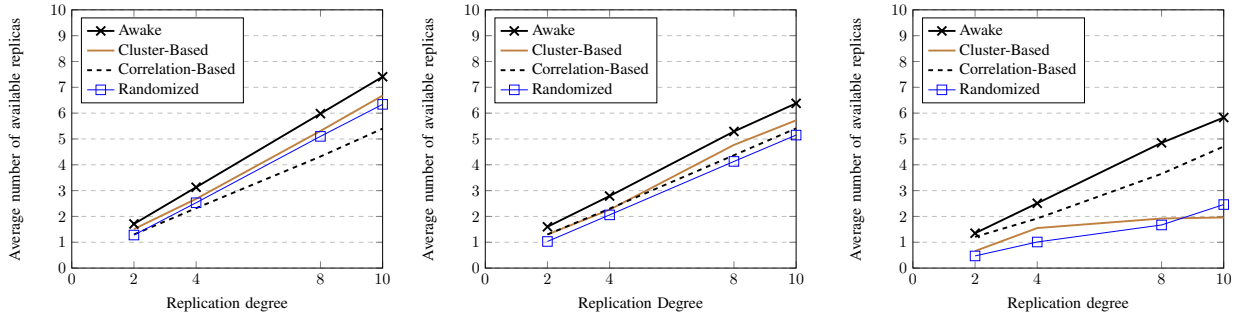



Fig. 4: Effect of the learning factor, $\beta$, on the performance of replication algorithms. Awake performs about 20% better compared to the best existing solution.




(a) High available churn model (b) Moderate available churn model (c) Low available churn model

Fig. 5: Average number of available replicas at each hour vs replication degree under different churn models. Compared to the best existing solutions Awake performs about 24%, 14% and 26% better under the high, moderate and low available churn models, respectively.

of randomized and cluster-based replication algorithms was verified with the higher replication degrees as well.

### C. Scalability

We examined all the replication algorithms under the moderate available churn model with the system sizes of 128, 256, 512 and 1024 nodes. In all of the simulation setups, the replication degree was fixed to 4 replicas. Figure 6 shows the scalability behavior of the replication algorithms when the replication degree is fixed and system size is scaled up. As shown in this figure, for a certain replication degree, the performance of all the replication algorithms of interest including Awake is independent of the system size. The same behavior was observed with the high and low available churn models. Also, our scalability analysis confirms that the results obtained in the 128 node scenarios are consistent with other system sizes as well.

Under the moderate available churn model and fixed replication degree of 4 replicas, on the average Awake outperforms the cluster-based replication algorithm as the best decentralized counterpart with the gain of about **23%**, where the average is taken over all the system sizes.

### D. Replication Time

A later replication time gives more chance to the data owner to learn about the availability behavior of the system. For this sake, we examined all the algorithms with the replication times of a day, two days, a week, and a month under the moderate available churn model in a system with 128 nodes and replication degree of 4 replicas. On average, about 1702 random transactions were initiated per *TS*.

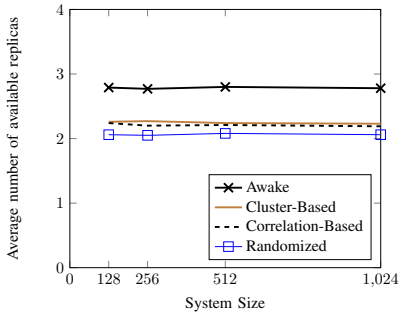



Fig. 6: Scalability of replication algorithms over different system sizes. Replication degree is set to 4. Awake performs about 23% better compared to the best previous work.

As shown in Figure 7, a replication time around two days -when each time slot has been updated at least once- is enough for all the replication algorithms to operate at their bests. A replication time longer than two days almost has the same effect on the performance of replication algorithms as a two days replication time does. The same behavior was observed with the high and low available churn models. Considering all the replication times, in comparison to the cluster-based replication as the best existing solution, on the average Awake performs around **21%** better.

### E. Running Time

Table V shows the average running time of Awake in seconds under the moderate available churn model as the system size is scaled up. The average was taken over 100 simulation runs. The running times were measured with the

| System Sizes (Nodes) | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|
| Running Time (Second) | 0.12 | 0.26 | 0.49 | 1.08 | 2.13 | 4.61 | 9.01 |
| Space Consumption (KB) | 3 | 6 | 12 | 24 | 48 | 96 | 192 |
| Communication Overhead (KB) | 0.37 | 0.43 | 0.48 | 0.57 | 0.59 | 0.64 | 0.71 |

TABLE V: Running time, space usage and communication overhead of Awake as the system size is scaled up. Replication degree is set to 4. Number of simulation runs = 100 times. Churn model = Moderate available model.

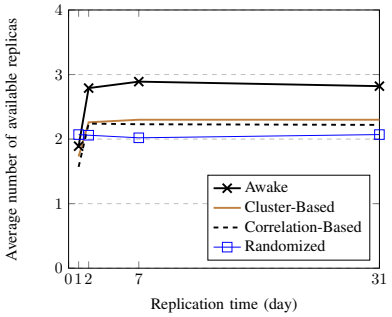



Fig. 7: The effect of replication time on the performance of replication algorithms. Awake performs about 21% better compared to the best existing solution.

system sizes of 128, 256, 512, 2048, 4096 and 8192 nodes. The replication degree in all of the simulations was fixed to 4 replicas. We also measured the running time of Awake under the high and low available churn models and similar results were obtained.

As shown in Table V, as the size of the system is scaled up, the running time of Awake increases linearly. At our simulation extreme with 8192 nodes system capacity, an execution of Awake takes around 9 seconds. Following the trend of running time, in a system with 1 million nodes, Awake is supposed to select the replicas in about 17 minutes.

### F. Space Consumption

In a system with $N$ nodes and availability vector of size $S$, the availability table of a node contains $N \times S$ probability values. Since $S$ is constant, the space complexity of Awake is $O(N)$ and hence is **linearly** dependent on the system capacity.

Empirically, in our simulations, the probability values were represented as positive integers between 0 to 100. We used the byte data type to cast the probability values. The space consumption of Awake as the system size is scaled up from 128 to 8192 nodes is presented in Table V. At our simulation extreme with 8192 nodes system capacity, Awake charges about 192 kilobytes on each node. Extending this trend of memory consumption, in a system 1 million nodes where the *TS* is 24, a node is supposed to need about 25 megabytes to store its availability table.

### G. Communication Complexity

In order to employ Awake on a structured P2P cloud storage, nodes need to piggyback their availability vector on the messages they route or initiate. The communication complexity is, therefore, proportional to the number of the nodes that the message is traversed on average during the routing procedure.

We employed a Skip Graph [28] as the underlying routing infrastructure for our P2P cloud storage. In a Skip Graph with $N$ nodes, a message traverses $O(\log N)$ nodes on average while it is routed from source to the destination. Hence, the communication overhead of employing Awake in such system is $O(S \times \log N)$ where $S$ is the size of nodes' availability vector. Since $S$ is constant, the communication overhead is simplified to $O(\log N)$ which is asymptotically the same as the number of nodes that a message is traversed on average during the routing procedure. Based on our simulation setup the communication overhead of Awake is **logarithmically** dependent on the system capacity.

The average communication overheads of Awake per message over the system sizes are represented in Table V. At our simulation extreme with 8192 nodes system capacity, an execution of Awake costs the communication overhead of about 0.71 kilobytes per message. Following the same trend, considering a system with 1 million nodes where the *TS* is 24, and by representing probability values with byte, Awake is presumed to charge the communication overhead of about 480 kilobytes per message on average.

## VII. Conclusions

To maximize the availability of replicas in the structured P2P cloud storages, we propose a novel availability aware replication algorithm. In order to select the replicas, our fully decentralized availability aware replication algorithm, Awake, employs the availability vectors of the nodes. The availability vectors are aggregated in a fully decentralized manner. The message overhead of this decentralized aggregation is asymptotically the same as the communication complexity of the structured P2P cloud storage that Awake is applied on. Awake has a linear space complexity in the number of registered nodes to the system. To the best of our knowledge, Awake is the only availability aware replication algorithm for the structured P2P cloud storages that maximizes the availability of replicas regardless of the underlying churn model of the system.

We extended the SkipSim simulator and simulated the state-of-the-art decentralized availability-based replication algorithms. In comparison to the best known decentralized existing solutions, on the average, Awake improves the availability of replicas with the gain of around **21%**. From the scalability point of view, as the system size is scaled up, Awake shows the same performance regardless of scaling. Awake only needs two days as the learning phase to learn the availability behavior of the system. In a system with **1 million** nodes, Awake is expected to select the replicas in about **17 minutes**, charge the space consumption of about **25 megabytes** on each node, and charge the communication overhead of about **480 kilobytes** on each message. As our future work, we plan to extend Awake to select the replicas based on the availability, location,

and storage load of nodes. Considering these factors together would likely result in the uniform distribution of replicas among high available nodes with minimum access latency to the data requesters.

## REFERENCES


[1] D. Yang, Y.-x. Zhang, H.-k. Zhang, T.-Y. Wu, and H.-C. Chao, "Multi-factors oriented study of p2p churn," *International Journal of Communication Systems*, 2009.

[2] R. Van Renesse and F. B. Schneider, "Chain replication for supporting high throughput and availability." in *OSDI*, 2004.

[3] C. Blake and R. Rodrigues, "High availability, scalable storage, dynamic peer networks: Pick two." in *HotOS*, 2003.

[4] P. Knežević, A. Wombacher, and T. Risse, "Dht-based self-adapting replication protocol for achieving high data availability," in *Advanced Internet Based Systems and Applications*. Springer, 2009.

[5] V. Simon, S. Monnet, M. Feuillet, P. Robert, and P. Sens, "Splad: scattering and placing data replicas to enhance long-term durability," *Technical Document, Inria*, 2014.

[6] H. Shen and C.-Z. Xu, "Locality-aware and churn-resilient load-balancing algorithms in structured peer-to-peer networks," *Parallel and Distributed Systems, IEEE Transactions on*, 2007.

[7] A. Datta and K. Aberer, "Internet-scale storage systems under churn–a study of the steady-state using markov models," in *P2P Computing*. IEEE, 2006.

[8] S. Ktari, M. Zoubert, A. Hecker, and H. Labiod, "Performance evaluation of replication strategies in dhts under churn," in *International conference on Mobile and ubiquitous multimedia, ACM*, 2007.

[9] S. Legtchenko, S. Monnet, P. Sens, and G. Muller, "Relaxdht: A churn-resilient replication strategy for peer-to-peer distributed hash-tables," *ACM TAAS*, 2012.

[10] J. Paiva, J. Leitao, and L. Rodrigues, "Rollerchain: A dht for efficient replication," in *NCA*. IEEE, 2013.

[11] M. Rahmani and M. Benchaıba, "A comparative study of replication schemes for structured p2p networks," in *9th International Conference on Internet and Web Applications and Services*, 2014.

[12] H. Shen, "An efficient and adaptive decentralized file replication algorithm in p2p file sharing systems," *Parallel and Distributed Systems, IEEE Transactions on*, 2010.

[13] M. Almashor, I. Khalil, Z. Tari, A. Y. Zomaya, and S. Sahni, "Enhancing availability in content delivery networks for mobile platforms," *Parallel and Distributed Systems, IEEE Transactions on*, 2015.

[14] S. Le Blond, F. Le Fessant, and E. Le Merrer, "Finding good partners in availability-aware p2p networks," in *Stabilization, Safety, and Security of Distributed Systems*. Springer, 2009.

[15] "Skipsim: https://github.com/yaahaanaa/skipsim."

[16] Y. Hassanzadeh-Nazarabadi, A. Kupcu, and O. Ozkasap, "Locality aware skip graph," in *ICDCSW*. IEEE, 2015.

[17] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *SIGCOMM*. ACM, 2006.

[18] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis, and modeling of bittorrent-like systems," in *ACM SIGCOMM*, 2005.

[19] R. Jiménez, F. Osmani, and B. Knutsson, "Connectivity properties of mainline bittorrent dht nodes," in *P2P Computing*. IEEE, 2009.

[20] J. L. J. Laredo, P. A. Castillo, A. M. Mora, J. J. Merelo, and C. Fernandes, "Resilience to churn of a peer-to-peer evolutionary algorithm," *International Journal of High Performance Systems Architecture*, 2008.

[21] D. Wu, Y. Tian, K.-W. Ng, and A. Datta, "Stochastic analysis of the interplay between object maintenance and churn," *Computer communications, Elsevier*, 2008.

[22] Z. Ou, E. Harjula, and M. Ylianttila, "Effects of different churn models on the performance of structured peer-to-peer networks," in *Personal, Indoor and Mobile Radio Communications*. IEEE, 2009.

[23] Z. Yang, Y. Xing, F. Xiao, Z. Qu, X. Li, and Y. Dai, "Exploring peer heterogeneity: Towards understanding and application," in *P2P Computing*. IEEE, 2011.

[24] S. Wakayama, S. Ozaki *et al.*, "A design for distributed backup and migration of distributed hash tables," in *Applications and the Internet, 2008. SAINT 2008. International Symposium on*. IEEE.

[25] A. Kermarrec, E. L. Merrer, G. Straub, and A. Van Kempen, "Availability-based methods for distributed storage systems," in *IEEE SRDS 2012*.

[26] J. Paiva and L. Rodrigues, "Policies for efficient data replication in p2p systems," in *IEEE ICPADS 2013*.

[27] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2002.

[28] J. Aspnes and G. Shah, "Skip graphs," *ACM TALG*, 2007.