Replica Placement Algorithm based on Peer Availability for P2P Storage Systems

Gyuwon Song*‡, Suhyun Kim‡, Daeil Seo*‡ and Sunghwan Jang*‡
Human Computer Interaction and Robotics Department, Imaging Media Research Center‡*
University of Science and Technology, Korea Institute of Science and Technology‡*
Seoul, Korea
{*sharp81, suhyun.kim, xdesktop, jangc*}*@imrc.kist.re.kr*

***Abstract*—Peer-to-peer (P2P) technology is an emerging approach to overcoming the limitations of the traditional client server architecture. However, building a highly available P2P system is quite challenging, in particular a P2P storage system. The reason is due to the fundamental nature of P2P systems: peers can join and leave at any time without any notice. Replication is one of the strategies in overcoming the unpredictable behavior of peers. A good replication algorithm should use the minimum number of replicas to provide the desired availability of data. The popular approach in the previous studies is a random placement of replicas, but it ignores the wide difference in the availability of each peer. In this paper, we propose PAT (Peer Availability Table) in order to analyze and predict the state of nodes and develop a replica placement algorithm, which exploits the availability pattern of each individual peer. By comparing our algorithm with a random placement scheme, we show that our algorithm dramatically improves the data availability with moderate overhead in terms of memory consumption and processing time in both ideal and practical conditions. Additionally, we demonstrate the application of PAT as an analysis tool for various P2P systems.**

***Keywords*-Peer-to-Peer storage system, replica placement, peer model, availability, BitTorrent**

I. INTRODUCTION

Peer-to-peer (P2P) technology is an emerging approach to overcoming the limitations of the traditional client-server architecture. P2P systems can provide high scalability and reliability by using peers' donated resources, including computing power, network bandwidth, and disk space. The P2P storage system focuses on storage service among the P2P systems. A previous version of this paper is published in [1], in which a replication scheme to build a highly available P2P storage systems. P2P storage systems such as OceanStore [2], Farsite [3], Freenet [4], and PAST [5] take advantage of the rapid growth of network bandwidth and disk size to provide persistent storage without central servers. Unlike P2P file sharing systems, P2P storage systems provide the functions that not only read data but also write, as does a traditional storage system. In this system, all data should be stored at other nodes redundantly because data could be corrupted. Also, data should be encrypted to guard the user's privacy because we cannot trust every peer. The users in the P2P storage system can access their data anytime, anywhere even though the computer that has worked with that data recently may be offline. Another advantage of the P2P storage system is that we have seemingly limitless storage space by using other nodes' storage. Namely, we can use the P2P storage system for storing data at other nodes' disks and then retrieve that data on demand. Furthermore, once data is written to the P2P storage system, that data is replicated and distributed to other nodes automatically. Thus, we can use the P2P storage system as an automatic backup solution.

However, building a highly available P2P system is quite challenging, in particular a P2P storage system. The reason is due to the fundamental nature of the P2P systems: peers can join and leave at any time without any notice. In other words, peers are not always available. Moreover, each and all of the availabilities are diverse. To make it clear, we set the definition of the $availability$ to be defined as "The degree to which a system, subsystem, or equipment is operable and in a committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random, time" [6]. The various availabilities of peers are important keys to improving the data availability in the P2P storage system. Assume that several peers are not available at the moment. Then, generally the data availability would be decreased. At this point, however, the data availability could improve if the remaining available peers take the responsibilities for storing and providing the required data as a substitute for unavailable peers. In order to this, the popular approach in recent studies is the random placement of the replicas, but that ignores the important properties of peers in the P2P storage system. The motivation of our research, millions of nodes have the potential to improve availability, is a highly available P2P storage system.

This paper argues a replica placement algorithm to enhance the data availability of the P2P storage system. The main idea is that select nodes, which have the most different availability but reasonable one among peers for replica storing. To evaluate the difference between peers' availability, we make PAT (Peer Availability Table) to represent a peer's availability. PAT is automatically managed by using a DHT-based P2P system. This novel algorithm can maximize the data availability efficiently with minor overhead.

The rest of this paper is organized as follows: in section II we describe a survey of related work; after the description

of the measurements and analysis of host availability results in section III, we propose our algorithm in section IV and its simulation evaluation in section V. In section VI, applications of our peer model are presented; and finally, we discuss the limitation of our work, and conclude this paper in section VIII.

II. Related work

There are many kinds of P2P systems such as file sharing, computing resource sharing, VoIP, and so on. Among them Farsite[3], Freenet [4], OceanStore [2], FreeHaven [7], Eternity [8], and PAST [5] are global storage systems intended for providing the scalability and self-organization of systems with persistence and reliability. [4] [7] [8] are more focused on the anonymity of users and anti-censorship for contents sharing. [3] is a server-less distributed storage system that has traditional storage system semantics. A directory group is used to ensure that the files are never lost. It showed that data is never lost as the maximum size of the clients is at the order of 10^5. Following this, we chose the size of the peer availability measurement. [2] provides a global persistent storage service that supports updates on replicated data. It uses erasure coding to provide redundancy without the overhead of strict replication and is designed to have a very long Mean-Time-To-Data-Loss (MTTDL). Also, [5] is a simple storage service for persistent, immutable files. It uses randomization to ensure diversity in the set of nodes that store a file's replicas and to provide load balancing. [9] [10] [11] described the replication method based on location, data consistency.

In order to improve the availability, a measurement of peer availability has to be done first. Since BitTorrent [12] has become the most popular among P2P file sharing systems, currently, there are many studies of measurements and analysis on BitTorrent systems. Izal et al. [13] analyzed a five-month workload of a single BitTorrent system for software distribution with thousands of peers, and evaluated the performance of BitTorrent at the flash crowd period. Bellissimo et al. [14] analyzed the BitTorrent traffic of thousands of torrents over a two-month period regarding the shared file characteristics and client access characteristics. Guo et al. [15] provided an understanding of torrent evolution in the BitTorrent systems and the relation among multiple torrents over the Internet. Measurements [16] [17] [18] characterize the P2P file sharing system's traffic over the Internet, including Napster, Gnutella, and KaZaa systems. But, they mainly focused on the performance of those systems. To easily measure a P2P network, Global Internet Measurement System (GIMS) was proposed in [19].

Some redundancy schemes are proposed to improve the data availability. In [3], their simulation showed that the random replica placement is better than the replacements that consider availability, due to the fairness of nodes' load. However, it was designed to support typical desktop workloads in academic and corporate environments. Because of this, nodes are less dynamic than in real P2P environments. TotalRecall [20] also uses a random placement scheme and proposed a static model for estimating data availability. They evaluated the data availability by the mean availability of peers as a parameter. Bhagwan et al. [21] and Blake et al. [22] used this scheme in the same manner. It is not a practical approach because they use a static model to evaluate host availability. Tian et al. [23] [24] studied the dynamic pattern of the Maze system and proposed a similar-MTTF-MTTR data placement scheme under the time-related model of data availability. They suggested the data availability should be considered as not a constant probability model, but a time-related probability model. [25] also pointed out that which uses the information of the session time to prevent the burst failures.

[3] [26] use the Hill-Climbing algorithm, which uses peers with high availability to replace peers with low availability. They are based on the host availability measurement to improve the data availability. Our research started from this concept. However, it is hard to analyze the host availability under this scheme since there is no availability model to measure it.

III. Measurement and Analysis of Availability

In this section, we describe a measurement and its analysis of the peer availability. We indentify the peer availability with the host availability and mainly use the former in this paper. The choice of the P2P system to measure the peer availability is quite an important decision because the results of measurement and analysis could be easily biased if we chose a P2P system which is popular in a specific country or is used for academic or research infra such as [27] [28]. So, we chose BitTorrent [12] as a representative P2P system. BitTorrent is a P2P file sharing protocol and has recently been showing strong growth. Usage of the protocol accounts for significant Internet traffic, though the precise amount has proven difficult to measure. According to a report [29], BitTorrent traffic occupied 53% of all P2P traffic on the Internet in June 2004. There are millions of simultaneous users and various client applications, as well. BitTorrent-like systems work in the following manner. The content provider creates a $.torrent$ (meta-data) file for content sharing, and publishes that file on a public web site or tracker web site. For each torrent file, there is a tracker site, whose URL is encoded in the torrent file, to find peers with whom to exchange the file chunks. The content provider then runs a BitTorrent client with a complete file to share as a seed. Another user who wants to download the file starts a BitTorrent client with the $.torrent$ file as a leech. After a handshake process, data transferring begins. Since any node can join or leave at anytime, data availability is highly reliant on the arrival and departure of peers in particular seed nodes.

A. Methodology

Our measurements have some assumptions: 1) BitTorrent users represent the other users in most P2P systems very well. 2) BitTorrent users keep running a BitTorrent client when they are using their computers. 3) There is no hacked client that is designed to not respond to a BitTorrent handshake. 4) Network addresses of the users in BitTorrent are not changed once assigned, so it can be used as an identifier. Based on this, we intensively measured the availability with the numerous peers in the BitTorrent system. We did not consider the tracker sites or system performances to up/download files because we were mainly focusing on the peer availability.

Protocol	BitTorrent v1.0
Date of measurement period	October, 2008
Duration of measurement	Oct.6 Oct.20 (2 weeks)
Measurement interval	Every 5 minutes
Measurement methodology	BitTorrent Handshake
Number of torrent files	100
Number of peers	96,749
Result record sets	492,829,138

Table I
SUMMARY OF THE MEASUREMENTS

Table I outlines our measurements. To analyze the peer availability of BitTorrent users, we gathered a large size of BitTorrent clients' log files and then extracted a list of the IP addresses and port numbers. Popular tracker sites such as Suprnova.org, Piratebay.org, and Movierg.com were used to obtain one hundred torrent files. Those torrent files had held a top 10 rank just before our measurement in Audio, Video, Application, Game, and Other categories. The number that we collected of distinct peers added up to 96,749.

We devised a multi-thread crawler to evaluate the peer availability concurrently. Since the crawler does not participate in swarm for transferring contents, no file chunk was transferred. The crawler tries to establish a connection (BitTorrent handshake) with the listed peers every 5 minutes and then inserts the results into the database. Inserted records total more than 490 million and all peers have exactly 4,032 records after filtering out. It means no errors occurred during our measurement.

B. Result Analysis

Geographical distribution Table II shows the geographical distribution of peers that are piled in our measurement. We extracted the geographical distribution from the users' IP addresses using MaxMind GeoIP [30].

North america holds the foremost position among other countries and Europe is the second. The interesting thing is that peers in BitTorrent are distributed world-wide in 191 countries, however distribution is not even but dominated by just three countries(US, UK, and Canada).

Rank	Country	# of peers	Percent
1	United States	22,532	23.29%
2	United Kingdom	9,043	9.35%
3	Canada	7,534	7.79%
4	Australia	4,168	4.31%
5	Sweden	3,091	3.19%
6	Poland	2,984	3.08%
7	Brazil	2,681	2.77%
8	France	2,669	2.76%
9	Netherlands	2,075	2.14%
10	India	2,035	2.10%
11	Norway	1,946	2.01%
12	Spain	1,767	1.83%
13	Portugal	1,587	1.64%
14	Philippines	1,581	1.63%
15	Germany	1,538	1.59%
16	Italy	1,516	1.57%
17	Greece	1,509	1.56%
18	Malaysia	1,457	1.51%
19	Korea	1,300	1.34%
20	Finland	1,253	1.30%
Others	$< 1\%$	22,483	23.24%
Total	191 countries	96,749	100%

Table II
GEOGRAPHICAL DISTRIBUTION OF PEERS

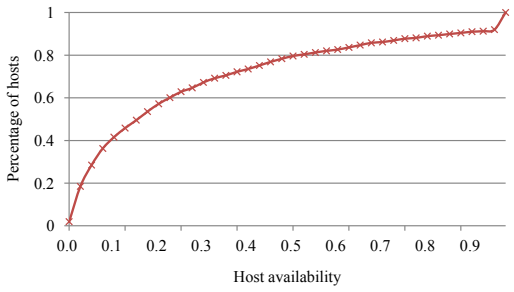

Figure 1. Cumulative distribution of the host availability

Peer availability We plot the cumulative distribution of the peer availability in Figure 1. The average of the availability is 28.39% and the median is 15.67%. It indicates that about 90% of the peers were not available which means only less than 10,000 peers frequently joined to BitTorrent during the two weeks of our measurement. Since neither errors with our crawler system nor network occurred while probing, probably BitTorrent users have a temporary usage pattern and serious lower availability. We deduce possible causes from this result. The first is a network related problem such as DHCP, firewall or NAT. Firewall and NAT do not allow the crawler to reach the peers which are behind them. Moreover, a peer's network address may be changed when it is newly leased from the DHCP server. [28] also pointed out limitations of which all these approaches rely on IP addresses. Probing by IP address does not accurately capture the available characteristics of the hosts. IP address probing would consider each IP address a new host, thus

greatly overestimating the number of hosts in the system and underestimating their availability. Using a unique identifier for a BitTorrent user could solve that, but most BitTorrent systems, including trackers and clients, do not support that in order to avoid an invasion of privacy or legal issues. The second possibility is derived from a specific usage pattern of BitTorrent users. Once a downloading process is completed, users do not run a BitTorrent client until they need to download another file. Basically, users in the P2P system tend toward selfishness; they want to use other resources freely as a free-rider and to avoid uploading what they have already downloaded from others. To allure users to share their surplus resources, a compensation mechanism is indispensably required.

On the other hand, about 0.9% of peers keep a ready-available state such as a server. Those peers appear at the right side of the curve in Figure 1. In all probability they use an exclusive BitTorrent machine all day to share contents. To improve the data availability, P2P storage systems should utilize these altruistic peers. Overall, the result of our measurement is similar to [23] [28] with respect to the host availability. And, some peers have diurnal online patterns but high available peers do not [23]. We have been maintaining the host availability measurement for long periods of analysis.

Time-related variation of peer availability Figure 2 presents the variations of three peers' availability as time passes. Three peers were sampled among those who had 28% availability which is the mean availability of the whole of the peers and those are distinguished by red, green, and blue. The circle in Figure 2 represents a wall clock (for 24 h) and a length of the radius refers to the availability. Namely, the red peer is highly available from 03:00 - 15:00, the green peer from 18:00 - 05:00, and the blue peer from 14:00 - 21:00. Even though they have the same average availability, their time-related variation of peer availability is conspicuously changed as time passes during the day. Here is a significant chance to improve the data availability. Previous works have a simple model to represent the peer availability, such as a static model. It assumed all peers have a static availability in the long term but it will change in the real P2P environment, as shown in the graph. It implies that we can maximize the data availability by selecting well-timed peers for failure tolerance. If we chose those three peers for replica storing, then we can access the data at anytime with high probability. However, assuming that we randomly, by the previous works' policy, choose some peers who have a static mean availability, $p = 0.3$. Then the availability is 90% when 12:00 15:00 and the rest of time's availability is nearly 0%. Possibly it is hard to access to the data except when 12:00 - 15:00. This point is the motivation of our research.

Other results If we extend the online pattern window from a day to a week, we can find the diverse distribution of availability along with, not only time of day, but also day of the week. For example, a five-day a week worker's home PC would not be used during the weekdays but be used mainly on the weekends. Also, the cultural special days which lead people to go out, such as Thursdays for a shopping day in Australia, make the weekly pattern more distinct.

IV. Replica Placement algorithm

After considering the factors in the previous section, we reached these preliminary conclusions. Many peers in the P2P system have low availability. Their availabilities are dynamically changed as time passes, not only the time of a day but the day of the week. In this environment, i.e., peers can join and leave the system at any time and node failure is no longer an exceptional event, but is common. Even though the P2P systems employ some schemes of data redundancy to recover unavailable data, it remains unclear what availability guarantees can be made using existing systems, or conversely how to best achieve a desired level of availability using the mechanisms available [21]. In this section, we describe the details of our system for replica placement. In particular, we propose a probabilistic model to represent the peer availability and a novel algorithm to improve the data availability based on our peer model for a highly available P2P storage system. Since we assume our system is built on a DHT system as [2] [5], it follows the DHT system's protocol.

A. Replication schemes

The goal of the replication is to use the minimum number of replicas to provide a high availability of data. To do this, both file replication and erasure coding schemes are generally used in a P2P system. File replication is a simple strategy that makes n copies of the file and puts them on different hosts. However, reproducing an entire file that is not accessible can be a burden in both storage space and time. Block-level replication makes it somewhat better, but if any single block cannot be found then the entire file object is useless. Erasure coding [31] provides the property that a set of b original blocks can be reconstructed from any m coded blocks taken from a set of cb (where m is typically close to b, and c is typically a small constant). Then, m coded blocks are stored at different hosts. As mentioned above, the key idea of the replication is that it makes redundancies of the original file and distributes them over other hosts for failure tolerance. In these procedures, we are mainly focusing on the 'other hosts', i.e., the best place to store redundancies of the file in order to maximize the data availability.

B. Peer Availability Table

Peer Availability Table (PAT) is a probabilistic model which represents peer availability. We designed a peer availability table and its manipulation methods. PAT indicates a

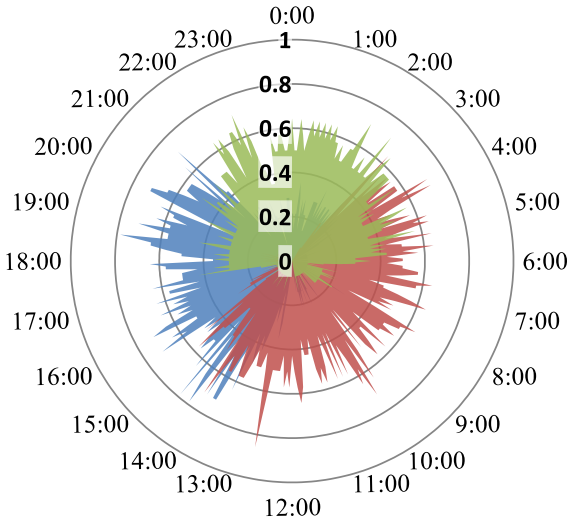

Figure 2. Host availability distribution with time of day for three sampled peers, which have the mean availability

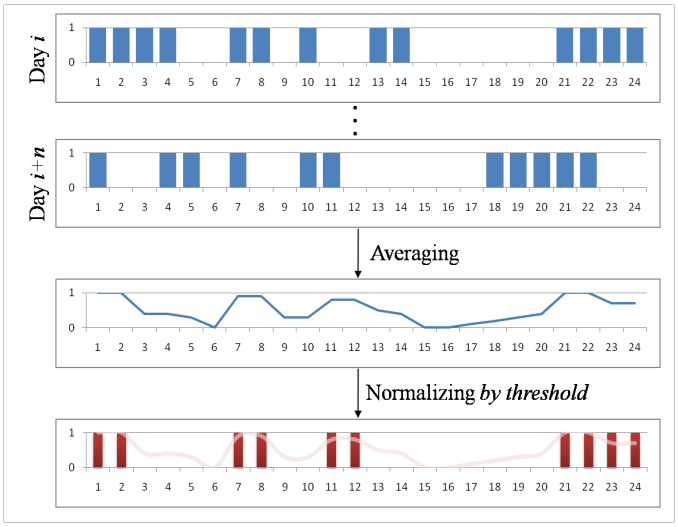

Figure 3. Process of building a Peer Availability Table

peer's availability at every five minute interval during a one-week period. A time slot is a basic unit to discretely divide the stream of time. The length of a time slot is five minutes, the same as our measurement interval. So an hour has 12 time slots, a day has 288, and a week has 2,016. We define the time slot 0 as Monday 0:00 and the last one as Sunday 23:55. The availability with time slot i is calculated by

$$AV_i = \frac{OnlineCounts}{VerificationCounts}$$

PAT can reflect the diverse aspects of characteristics of a peer. Assume that user A uses his office PC at the work place during the daytime on weekdays but uses his home PC at home all day long on the weekends (i.e., User A rarely uses his home PC on weekdays and his office PC on weekends). If PAT includes just daily availability, the availabilities of these two PCs would be underestimated by a mean value even if some days' availabilities are relatively high which it cannot neglecte to utilize. Besides, the cultural special days, which incur less use for a P2P system than other days, such as Thursdays for a shopping day in Australia,

are well reflected. More importantly is that since PAT has the information of the weekly availability, the P2P storage system can acknowledge the long-term availability of a peer that uses it for detecting a permanent leave of the system. The threshold time to decide a peer's permanent leave can be assigned from several hours to seven days. For example, suppose a peer's PAT is as follows:

Time Solot	0	1	2	...	2015
Verification count	28	20	17	...	25
Online count	21	17	17	...	20

Table III
AN EXAMPLE OF PAT

Table III refers that the peer who has 0.75 availability on Monday 0:00 - 0:04 and 0.8 availability on Sunday 23:55 - 23:59. After some weeks, if all availability values converge to zero for every time slot, the system can then perceive that this peer leaves the system permanently and starts a recovery process for inaccessible data. The $VerificationCounts$ can be different from others. Because there is no central server to evaluate all peers' availability in the P2P system, all peers have to evaluate partially along with neighbor peers and by themselves. Each peer manages a personal PAT and neighbors' PATs only when they are online. Since peers independently join and leave the system and only verify when they are online, the $VerificationCounts$ can be different with each other's neighbors. The process of PAT management is as follows. A peer's PAT is initialized with all values at zero for the first time bootstrapping to the P2P storage system. During the bootstrapping step, the peer gets a routing table that includes neighbor nodes lists. According to the list, the peer creates PATs for neighbor nodes. The peer's PAT is actively updated every 5 minutes by itself, while the neighbor nodes' PATs are updated passively. The passive method is based on a heartbeat message. All nodes in DHT-based systems send a heartbeat message (or keep-alive message) to their neighbor nodes to inform them that a node is available periodically. Peers who receive a heartbeat message from another node increase the $VerificationCounts$ and the $Onlinecounts$ of the sender's PAT at the correspondence time slot to the received time. Following these processes, all nodes' PATs are maintained up-to-date. Since a user's usage pattern may be changed over time, clearly, the PAT should be renewable in a few weeks or months. This is a topic for future consideration.

C. The Similarity of PAT

According to section III, peers are very dynamic (i.e., each peer has a diverse PAT), but we can classify peers by the degree of similarity of PATs. Because there are a large number of peers in the P2P system, it is probable that there exist some peers who have a similar PAT. The degree of similarity between peer A and peer B is calculated by

$Sim_{A,B} = \Sigma P_{A_i} * P_{B_i}$, where P_{A_i} is available at time slot i on peer A's PAT, and P_{B_i} is respectively. A high degree of similarity between peer A and peer B means that their usage patterns are very much alike, while a very low degree of similarity means they use the system oppositely from each other. Our novel algorithm for replica placement begins at this point. Choosing peers who have a high degree of similarity ensures a highly available P2P storage system when trying to access data at a usual usage time for the owner. On the other hand, choosing peers who have a low degree of similarity also ensures even when trying to access data at quite a different time from his normal usage pattern. Note that when selecting nodes, these nodes' average availabilities must be greater than the threshold to guarantee minimum availability since the similarity will be zero if calculated with a peer whose availability converges to zero. Therefore, if the degrees of similarity were less than a threshold, that combination would be discarded.

D. Replica Placement Algorithm

In this section, we describe how our algorithm works for replica placement based on PAT. To build a highly available P2P storage system, all data must be stored redundantly. The core algorithm, to be brief, is that it finds the best combinations among peers who have various similarities of PAT to maximize the data availability. We refer to the chosen peers who maximize the data availability as a max_list. The size of a max_list is decided by a given parameter as target availability, so it would minimize overhead that is related to network bandwidth and storage space. The following pseudo-code is our algorithm to construct a max_list. The parameters of this method are a $file\_id$ that wants to store and target availability that represents how important it is. In P2P storage systems such as [5], mostly, each node is assigned a 128-bit node_id, derived from a cryptographic hash of the node's public key. Each file is assigned a larger bits $file\_id$ rather than a $node\_id$. When a file is inserted into the system, some nodes are selected whose $node\_id$ are numerically closest to the 128 most significant bits of the $file\_id$. The $candidates\_list$ is compiled of those nodes and is relatively long. To filter out useless nodes in the $candidates\_list$, similarities are calculated with the host user's PAT and organized into the $similar\_list$. This procedure would reduce the processing cost of node filtering. Composing a max_list is as follows. A peer who has the highest similarity value is added in the max_list. Then, generates all combinations of nodes, i.e., ${}_nC_r$, where n is the size of the $similar\_list$ and r is $n-1$, and calculates the availability with their PATs. The data availability is evaluated by integrating the probability density function of the peer's PAT. The loop will be terminated if evaluated availability is greater then the $target\_availability$. Finally, the combination nodes that have the highest data availability construct the max_list. Replicas are placed to nodes according to the

max_list.

Algorithm 1 MaxList($fileId, target\_availability$)

```
candidates_list ⇐ nodes for given file_id
for all each node n in candidates_list do
  similarity ⇐ calc similarity (n, me)
  if similarity > threshold then
    similar_list ⇐ add node n
  end if
end for

for i = 0 to similar_list.size do
  data_availability_list ⇐ calc dataAvailability with
  all combinations (similar_list.size, i)

  sort list order by data_availability
  if data_availability > target_availability then
    break
  end if
end for

max_list ⇐ the top entry of the data_availability_list
return max_list
```

V. Simulation evaluation

We designed a simulation environment to evaluate our new algorithm. The simulator is based on a P2P storage system, which is built upon a DHT system. We detached a part that is node selecting from the system and simulate this part only. Even though we do not simulate a whole P2P storage system, the result of our simulation is worthy because we use real trace data that we measured which evaluates data availability in our simulation by replaying that data. We explore the improvement of data availability by our novel algorithm and compare it with the random placement algorithm. Our simulation is focusing on how to achieve the best data availability using replication schemes with low overhead. All simulations were performed 100 times with various simulation settings under an Intel Core Duo 2GHz/2GB RAM PC.

A. Assumptions

Actually, there may be hundreds of thousands of users in a P2P system. Since it is impossible to simulate the whole size of the P2P system, 1,000 peers were randomly sampled among all peers that we had measured for the peer availability. As we mentioned in section III, peers have various availabilities and its average is very low. Douceur et al. [32] reported that about 60% of the nodes in Napster and Gnutella systems had less than 70% availability, due mainly to the dynamic behavior. Moreover, our measurement of the BitTorrent system shows that about 80% of the nodes have less than 50% availability and the top 10% of the nodes have more than 90% availability. Therefore, the modeling of peers for a simulation is a critical factor of the data availability measurement since it is one of the immediate causes. We made peer models of three types: a dynamic peer who cyclically turns on and off the system; a server-like peer who is always online; and, an inactive peer who rarely uses the system, i.e., $p < 0.1$. To make a worsecase scenario, we assume no server-like peers exist in our simulation but inactive peers are considered from 0% to 75%. In our simulation, all used PAT data are real measured data during a two-week period and we evaluated the data availability by replaying its results. The evaluation ran a hundred times during a one-week period for each data. To simplify our simulation has some limitations; the network environment is error free and all data transmission is completed at once. The size of the $candidates\_list$ and $target\_availability$ are given as a parameter. In future work, we consider the whole simulation of a P2P storage system in a real network based on long-term measurement analysis.

B. Simulation Results

Our simulation results are divided into three topics: the improvement of data availability, the overhead of processing cost and memory usage, and the differences with respect to the change of configuration settings. We measured the data availability improvements and the processing cost on the ideal condition in which there are no inactive peers. In this condition we only changed the size of the $candidates\_list(n)$ and the $max\_list(k)$, which is referred as ${}_nC_k$. Note that the actual size of the max_list is fixed to fulfill the $target\_availability$. However, to compare more clearly, we input that as a parameter. We also observed the overhead of our algorithm with respect to memory usage. Lastly, we evaluated the effects of the ratio of peer models between a dynamic peer and inactive peer. Studies show that most peers have very low availability in our measurements as well as prior works. So, the configuration schemes of the inactive peer ratio were set as 0% (ideal), 10%, 30%, and 50%. In this simulation, we treat a peer whose availability is less than 10% for two weeks as the inactive peer.

We plot the main results of our simulation in Figure 4. Using our new algorithm can reduce the number of replicas of a file by half rather than a random placement in order to provide the data availability of over the 99.99%. Since the size of multimedia contents has been increasing recently, the resource usage is seriously important in both network bandwidth and disk space. Therefore, the result means that our algorithm can greatly improve efficiency of the resource usage rather than a random placement scheme. The size of the $candidates\_list$ does not impact on the data availability. However, the processing cost rapidly increases along with its size, as shown in Figure 5. The processing time is just 1 (msec) to select nodes by random but increased from 90 to

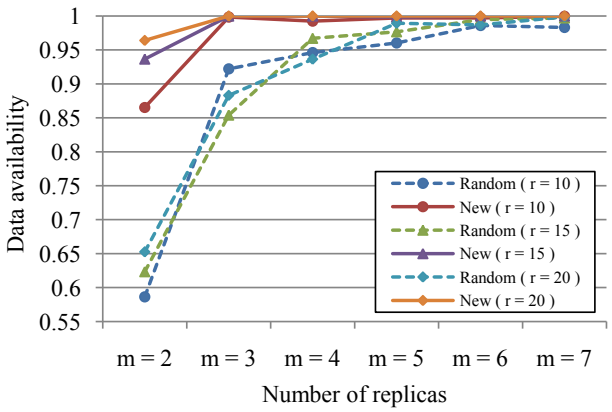

Figure 4. The data availability and required number of replicas

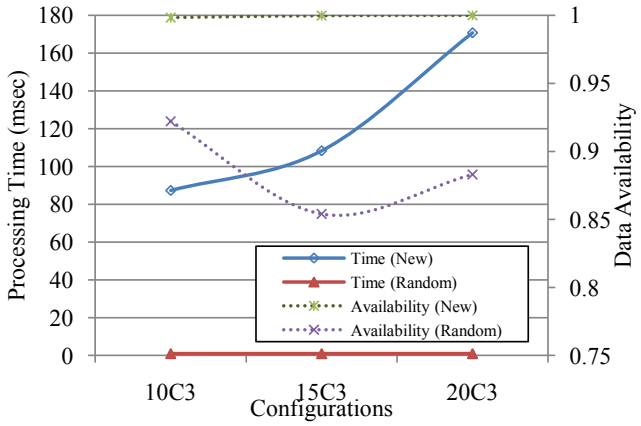

Figure 5. The processing time overhead with data availability

170 (msec) according to following configurations of ${}_{10}C_3$ to ${}_{20}C_3$. Considering the network latency time in real networks, a faster setting is much better but we think it is an affordable cost to get the high availability. We conclude that the size of the $candidates\_list$ is 15 which is sufficient to provide very high data availability when using our new algorithm. Figure 6 shows the memory usage for each algorithm and it includes all users of PATs that are used in our simulation and related objects to manage PATs. There is a slight difference between the two algorithms, about 1.5MB. In order to use our algorithm, the host PC uses more memory but most PCs are good enough to ignore that. Considering all these results, our replica placement algorithm can greatly improve the data availability with affordable overheads rather than random placement algorithm.

Finally, we evaluated the effects of the ratio of peer models between a dynamic peer and inactive peer. We set the ratio of inactive peer as 10%, 30%, and 50% and compared it with the ideal condition (0%). At this point, we use a ${}_{10}C_3$ scheme for our algorithm. Our simulation was designed to perform under more practical situations. Figure 7 shows the results. Though the ratio of the inactive peer increased from 10% to 50%, our algorithm provided over 90% data availability. On the other hand, the random placement scheme's data availability had fallen to under 65%.

In conclusion, we show that our algorithm can greatly improve the data availability while minimizing the waste of resources rather than random placement in ideal and practical conditions, as well.

VI. Application of Peer Availability Table

Peer Availability Table (PAT) is the core system of our new replica placement algorithm. Namely, the system, using PAT, logs peers' availability and give a clue to predict their usage patterns based on it. As we mentioned above section, PAT system is well suited to BitTorrent system. In this section, moreover, we show a feasibility study on PAT

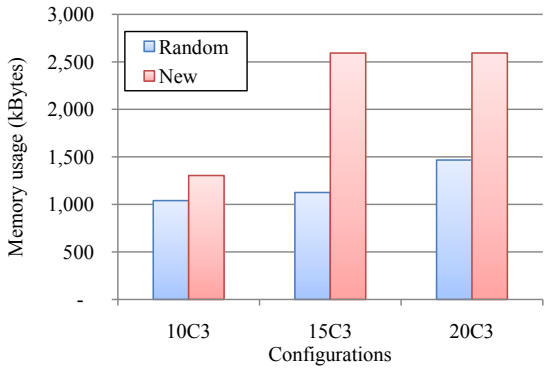

Figure 6. The comparison of the memory usage (Kbyte)

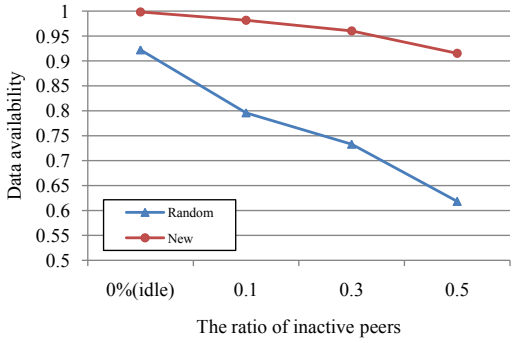

Figure 7. A change of the data availability with the ratio of inactive peers

system for other P2P systems. To explain it, we chose KAD trace [33] and Skpe trace [34].

The reason why we choose is that KAD trace has a huge log results in terms of number of peers and experiment period and has the measurement resolution same as ours. By the way, KAD is a Kademlia-based routing protocol [35] implemented by several peer-to-peer applications such as eMule and aMule which have a lot of simultaneous connected users. Skype, and then, which is P2P based VoIP system, has became a killer application of P2P systems recently. By the analysis of these trace data using PAT, we present an evidence that the feasibility of PAT as a tool to analyze of usage pattern for not only BitTorrent system but also any other P2P systems.

A. KAD

We sampled 12,241 peers which had checked as online for over 30 days among 400,375 whole peers. Though the original KAD trace has 5 minutes resolution of measurement, we adjust that to 1 hour since Skype trace has 1 hour resolution. Therefore the length of PAT was set 24 for a day and 168 for a week. Figure 8 shows the average of availability for a day and a week. The x-axis means time and the y-axis means the average of availability. The red line shows a daily pattern, the blue line shows a weekly pattern, and the green bar means the difference between two. Since a difference between a daily pattern and weekly pattern means over-/underestimation of availability, we have already pointed out that a daily/weekly pattern of peers clearly exists and a weekly pattern is more important in order to predict the usage pattern accurately. Specific days(Day1 and Day2) have a high availability rather than others(5 days), and then we can speculate that Day1 and Day2 are weekend, others are weekday.

B. Skype

Sky trace has total 4,000 nodes' result with 1 hour measurement resolution. Like as KAD, 2,081 super nodes of Skype trace are selected in this analysis. Interesting thing that the blue line decreases at Day5 and Day6 in figure 9. is the contrary result to KAD trace. Namely, two days' availabilities increase in KAD but decrease in Skype. In general, KAD is used to share file sharing but Skype is used to talk with acquaintances. Thus we can speculate that KAD is more employed at weekend for sharing files, however Skype is less employed at weekend for contacting through the Internet.

C. Usage pattern prediction

Based on PAT, we can predict nodes' states with high probability. In order to show the probability of prediction, we made a simple simulator. The simulator built PATs using the whole trace data of KAD and Skype except last one week result. And then, the simulator simply counts a difference a given PAT and a last one week log(which is excluded when building a PAT) of real trace. The result of prediction accuracy is plotted in figure 10. In KAD case, all results show a high accuracy ($> .9$) and the result which is using weekly PAT is slightly better than daily PAT. But the results of Skype are differ from KAD. The results which is using weekly PAT show $> .8$ stably, but the others which is using daily PAT go on worsening. This is another proof to illustrate that the weekly PAT is proper length.

VII. Discussion

As we described above, our novel algorithm can improve the data availability efficiently. However, there are some implementation issues in practical since it completely depends on availability logs of peers. Namely, all logs for each peer must be stored, and it is difficult to decide that who verify the status of a peer and how to share that logs. It remains an open problem to determine the best settings for real P2P environment.

VIII. Conclusion

In this paper, we studied an algorithm to improve the data availability in a P2P storage system. It is one of the most difficult topics in a peer-to-peer system. We showed that

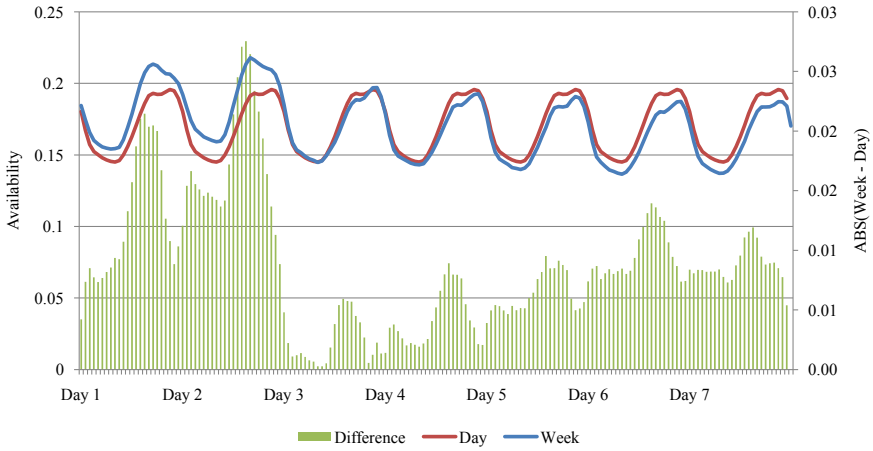

Figure 8. KAD trace

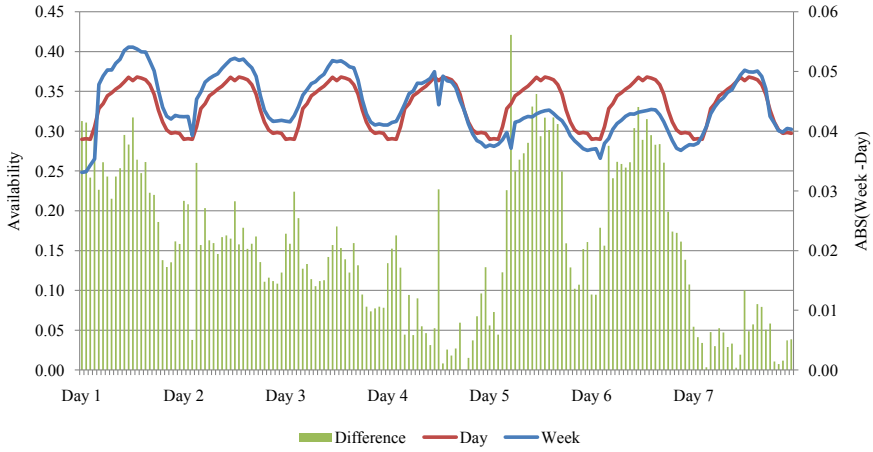

Figure 9. Skype trace

the peers' availability in a BitTorrent system as a result of our measurement. The results imply that the peer availability changed not only from time of day but also day of the week. The mean availability was relatively low due to the limitation of the IP based measurement.

Based on this result, we developed a probabilistic model, referred to as Peer Availability Table (PAT), representing a peer's weekly availability which means that it can cover the range from short-term availability to long-term availability in a simple manner. It can be used to find a peer who has similar usage patterns or detect permanent leave of a peer to failure recovery. We then propose a replica placement algorithm to maximize the data availability. To build a highly available P2P storage system, all data must be stored redundantly. The key was to make redundancies of the original files and distribute them over other hosts for failure tolerance. In these procedures, we mainly focused on the storage area of the file redundancies. Unlikely previous works that are a random placement scheme, our algorithm found the best combination of peers to provide the highest data availability among candidate peers. Because calculating data availability with all combinations is highly complex, we used a heuristic method to reduce the case of combinations by an estimation of similarity between PATs.

By comparing our algorithm with a random placement scheme, we showed that our algorithm dramatically improved the data availability with moderate overhead in terms of memory consumption and processing time in both ideal and practical conditions. Additionally, we demonstrate a feasibility of PAT as an analysis tool for P2P systems such

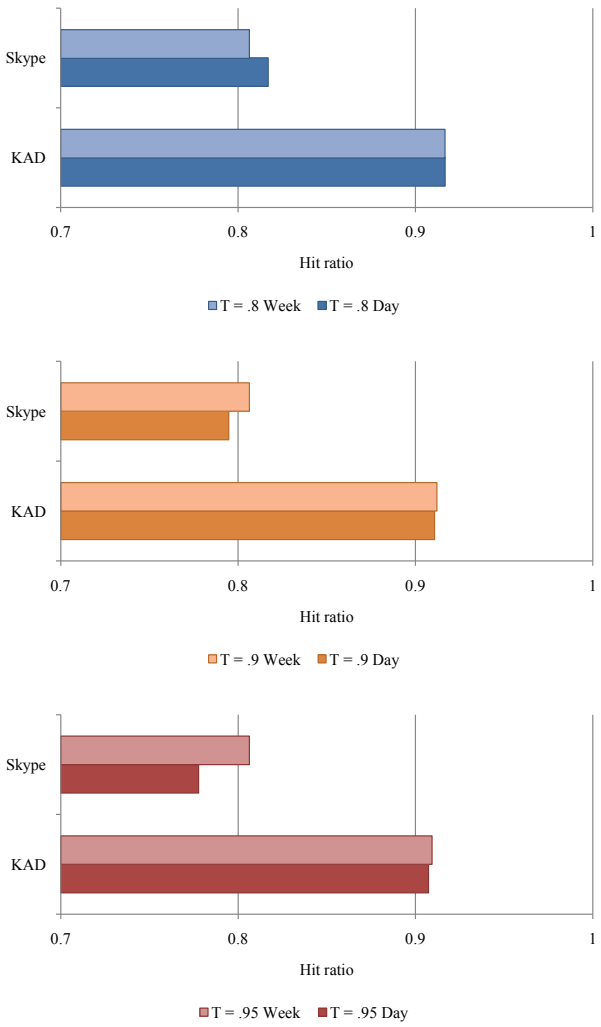

Figure 10. Prediction accuracy, T = threshold for building PAT

as KAD and Skype. It means that PAT is well suited not only BitTorrent system but also other P2P systems. Applying our new replica placement algorithm to other P2P systems and its verification in the real network are our ongoing works.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MKE/KEIT. [KI002119, Development of New Virtual Machine Specification and Technology]

REFERENCES

[1] G. Song, S. Kim, and D. Seo, "Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems," *Proceedings of First International Conference on Advances in P2P Systems*, pp. 160–167, 2009.

[2] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells *et al.*, "Oceanstore: An architecture for global-scale persistent storage," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 5, pp. 190–201, 2000.

[3] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," *OPERATING SYSTEMS REVIEW*, vol. 36, pp. 1–14, 2002.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 46–66, 2001.

[5] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 188–201, 2001.

[6] F. Standard, "1037c: Glossary of telecommunications terms," *The Institute for Telecommunication Sciences, Oct*, 2006.

[7] R. Dingledine, M. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 67–95, 2001.

[8] R. Anderson, "The eternity service," vol. 96, 1996, proceedings of Pragocrypt.

[9] J. Gossa, J. Pierson, and L. Brunie, "FReDi: Flexible Replicas Displacer," in *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, 2006, pp. 16–16.

[10] E. Sithole, G. Parr, S. McClean, and P. Dini, "Evaluating Global Optimisation for Data Grids using Replica Location Services," in *Networking and Services, 2006. ICNS'06. International conference on*, 2006, pp. 74–74.

[11] H. Yoshinaga, T. Tsuchiya, H. Sawano, and K. Koyanagi, "A Study on Scalable Object Replication Method for the Distributed Cooperative Storage System," in *Proceedings of the 2009 Fourth International Conference on Digital Telecommunications-Volume 00*. IEEE Computer Society, 2009, pp. 96–101.

[12] B. Cohen, "Incentives build robustness in bittorrent," vol. 6. Berkeley, CA, USA, 2003, workshop on Economics of Peer-to-Peer Systems.

[13] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 1–11, 2004.

[14] A. Bellissimo, B. Levine, and P. Shenoy, "Exploring the use of BitTorrent as the basis for a large trace repository," *University of Massachusetts, Tech. Rep*, 2004.

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

[16] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy, "An analysis of internet content delivery systems," *ACM SIGOPS Operating Systems Review*, vol. 36, no. si, p. 315, 2002.

[17] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," vol. 2002, 2002, proceedings of Multimedia Computing and Networking.

[18] P. Gummadi, S. Saroiu, and S. Gribble, "A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 82–82, 2002.

[19] D. Xu, T. Liu, D. Qian, Z. Luan, and M. Tang, "A New P2P-like Architecture for Large Scale End to End Network Measurement," in *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, 2006, pp. 62–62.

[20] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: System support for automated availability management," 2004.

[21] R. Bhagwan, S. Savage, and G. Voelker, "Replication strategies for highly available peer-to-peer storage systems," *Proceedings of Fu-DiCo: Future directions in Distributed Computing, Jun*, 2002.

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

[23] J. Tian and Y. Dai, "Understanding the dynamic of peer-to-peer systems," 2007, proc. of the 6th Int'l Workshop on Peer-to-Peer Systems.

[24] J. Tian, Z. Yang, and Y. Dai, "A data placement scheme with time-related model for p2p storages," 2007, pp. 151–158, peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on.

[25] K. Kim, "Time-related replication for p2p storage system," in *Proceedings of the Seventh International Conference on Networking*. IEEE Computer Society, 2008, pp. 351–356.

[26] T. Schwarz, Q. Xin, and E. Miller, "Availability in global peerto-peer storage systems," 2004.

[27] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs," *SIGMETRICS Perform. Eval. Rev.*, vol. 28, no. 1, pp. 34–43, 2000, 339345.

[28] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding availability," *Peer-to-Peer Systems Ii*, vol. 2735, pp. 256–267, 2003, kaashoek, F Stoica, I 2nd International Workshop on Peer-to-Peer Systems FEB 21-22, 2003 BERKELEY, CALIFORNIA.

[29] A. Parker, "The true picture of peer-to-peer filesharing," 2004.

[30] MaxMind, "Maxmind geoip country database."

[31] F. MacWilliams and N. Sloane, *The theory of error-correcting codes*. North-Holland Amsterdam, 1977.

[32] J. Douceur, "Is remote host availability governed by a universal law?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 3, pp. 25–29, 2003.

[33] M. Steiner, T. En Najjary, and E. Biersack, "Analyzing peer behavior in KAD," *Institut Eurecom*, 2007.

[34] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," in *Proc. of IPTPS*, vol. 6. Citeseer, 2006.

[35] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Proceedings of IPTPS02, Cambridge, USA*, vol. 1, pp. 2–2, 2002.