An Effective Replication Technique Using Rateless Codes for Unstructured P2P Networks

Keiichi Endo, Ryosuke Hamabe, Dai Okano, Kaname Amano
Graduate School of Science and Engineering, Ehime University, Matsuyama, Japan
Email: endo@cs.ehime-u.ac.jp

Abstract— **It is possible to recover the original data from a certain amount of encoded data by using rateless codes, which are utilized for multicast streaming. In this paper, we propose an effective replication technique using rateless codes for unstructured P2P networks. More specifically, we propose a method to spread and find chunks generated from files efficiently. Through simulation experiments, we show that the proposed technique achieves high search success ratio and shortens the time required to obtain a file.**

Index Terms— **unstructured P2P networks, content sharing systems, replication, rateless codes, erasure codes**

I. Introduction

Various Internet services are available today owing to the rapid growth of the Internet. Many of these services use the client/server (C/S) model in which a computer receiving the service (a client) is connected to a dedicated computer providing the service (the server). This model is based on the relationship between a master and a servant. The C/S model allows client data to be centrally managed on the server and thus has the advantage of being easy to design as a business model. In recent years, the rapid migration of everyday household connections to broadband has made it possible for such Internet services to transfer large volumes of content, and high-quality sophisticated services are expected in connection with these large-volume communications in the corporate world. It is extremely costly for the C/S model to provide such advanced services; the costs arise from the need to provide backup servers and to invest in equipment that can keep pace with the newest advances in processing capacity and can support an increasing number of clients. Quite simply, it is becoming more and more difficult to provide stable services using the C/S model.

A new network model that is currently attracting much attention is the peer-to-peer (P2P) model. In this network model, computers (peers) on the network communicate and share files on an equal footing with each peer serving as both a server and a client. As a result, traffic is not concentrated only at the server unlike the case in the C/S model, thereby reducing the cost required for investing in new equipment.

Manuscript received March 14, 2012; accepted March 28, 2012.

This paper discusses the use of rateless codes for replica placement on P2P networks and the distributed placement of multiple chunks generated from files on network peers. We propose a technique to distribute and find chunks efficiently. Through a simulation that permits the participation and removal of peers, we demonstrate the superiority of our proposed technique in terms of the time it takes to efficiently search for and obtain files.

This paper is structured as follows. In Section II, we will outline the P2P model and discuss existing search methods and replica placement techniques used in unstructured P2P networks. In Section III, we will discuss erasure codes, with a focus on rateless codes. In Section IV, we will explain our proposed technique, and in Section V, we will present the results of our simulation experiments. Finally, in Section VI, we will summarize our research and discuss future research topics.

II. P2P model

The P2P model is structured such that peers communicate on an equal footing. Because each computer switches its role between a server and a client while exchanging information in this model, processing is not concentrated on a specific server as in the case of the C/S model. As a result, we can reduce the possibility of computer performance forming a bottleneck in the network, and provide improved defect resistance and traffic distribution.

P2P models are classified into two types: hybrid P2P models in which a server manages the entire network topology and index information, and pure P2P models in which there is no management server. In this paper, we concentrate only on pure P2P models because the management server may become a bottleneck in hybrid P2P models. Pure P2P models can be categorized into two: structured P2P models, which conduct the routing of search queries on an overlay network, and unstructured P2P models, which do not perform this kind of routing.

In structured P2P models, it is possible to search more efficiently by constructing a topology for data searches, which is based on a specific algorithm. Distributed hash tables (DHTs) [1] are often used in structured P2P network designs. Chord [2], content-addressable network (CAN) [3], Pastry [4], and Tapestry [5] are representative examples of DHTs.

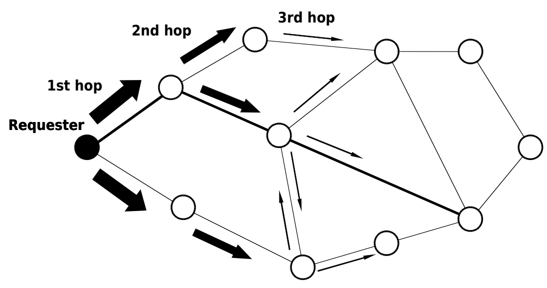


Figure 1. Example of flooding search.

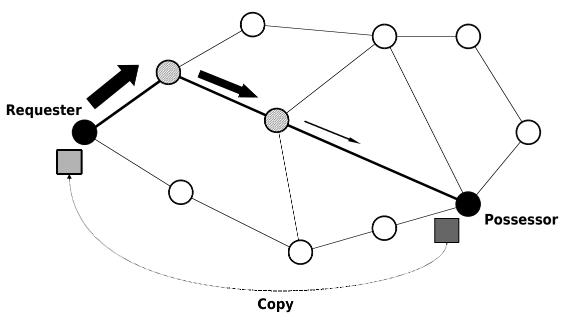


Figure 2. Example of owner replication.

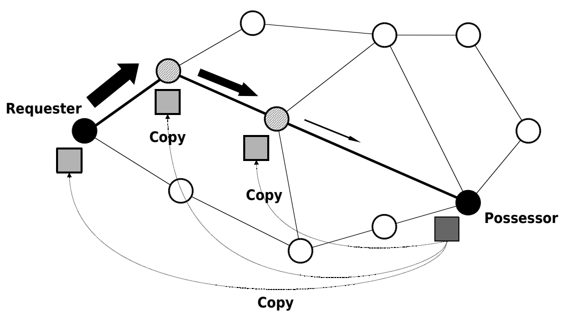


Figure 3. Example of path replication.

In contrast, unstructured P2P networks do not impose restrictions on the topology, file placement, or the size of the network. Our research contributes to the literature on unstructured P2P networks.

A. *Search methods for unstructured P2P networks*

Flooding [6] and k-walker random walk [7] are common search methods for unstructured P2P networks. We explain the flooding method in this subsection.

Flooding is a search method in which a peer that has a search request (the search request peer) sends a search request message (query) to all directly connected peers (adjacent peers), and each peer that receives the query in turn forwards the query to all its adjacent peers except the sender of the query (Figure 1). The actual procedure is described below.

1) The search request peer sends queries to all adjacent peers. The time-to-live (TTL) indicating the maximum number of transfers is set in these queries.
2) A peer receiving a query performs the following processes:
   a) If the peer has a file that matches the query, it sends a response message to the search request peer.
   b) The TTL of the query is reduced by 1.
   c) If the TTL is 1 or greater, the peer forwards the query to all adjacent peers other than the sender of the query, except that it does not forward any query that has previously been received.

An advantage of unstructured P2P networks is that they can perform flexible searches, i.e., searches in which it is possible that there are multiple files matching the query, such as partial match searches and full text searches. Flexible searches are difficult in structured P2P networks. Moreover, because there is a high degree of freedom in the network structure, unstructured P2P networks are also superior with respect to extensibility and fault resistance, and they are comparatively easy to implement. Disadvantages of unstructured P2P networks include the possibility of files disappearing from the network owing to the removal of peers and the possibility of failing to discover a file that actually exists in the network. We study techniques for placing replicated files on unstructured P2P networks to address these disadvantages.

B. *Replication technique for unstructured P2P networks*

Replication techniques resolve the aforementioned disadvantages of unstructured P2P networks by placing replicas of files on peers. We describe two such techniques.

1) Owner replication: In the owner replication technique [6], the file downloaded by the search request peer is published to the network in the same way as the original file. Since replicas are not placed on any other peers, files spread slowly. Figure 2 provides an example of replica placement using the owner replication technique.

2) Path replication: In the path replication technique [6], the file downloaded by the search request peer is published on the network in the same way as the original file, similar to the owner replication technique. Replicas of the target file are then placed on all the peers along the query forwarding route (search path) from the search request peer to the peer in which the target file is found, and the placed files are published on the network. Figure 3 provides an example of replica placement using the path replication technique.

The path replication technique allows a file to spread more quickly than the owner replication technique. However, high-degree peers (peers with many adjacent peers) have a high probability of appearing on search paths in this technique, which makes it more likely that replicas are placed on high-degree peers, thus increasing load on high-degree peers.

III. Erasure codes

In this section, we discuss erasure codes, which are error-correcting codes used in the proposed technique when creating chunks that are placed on a network after a target file is downloaded. Then, we discuss our research on several P2P systems that use erasure codes.

A. Reed-Solomon codes and rateless codes

The Reed-Solomon code [8] is an error-correcting code that uses Galois field arithmetic. This code is used in various fields, including DVDs and QR codes. However, a disadvantage of this code is that encoding and decoding calculations take a long time. Furthermore, when encoding, it is necessary to determine the number of redundant blocks to be added, and packet loss must therefore be estimated when this code is used for communications.

There have been several encoding formats proposed for rateless codes [9], in which it is not necessary to determine the number of generated encoded blocks. These rateless codes are used for applications such as multicast streaming [10]–[12]. When a sender continually multicasts encoded packets using rateless codes, the receiver can recover the original data when a certain amount of data is received, regardless of the quality of communication (the packet loss rate).

Luby transform (LT) codes [13] are a type of rateless codes. In LT codes, a degree d is initially chosen at random according to a probability distribution p_d $(d = 1, \dots, k)$. Next, d blocks are chosen from k data blocks at random and an encoded block is created using the EXCLUSIVE-OR of these blocks. By repeating this procedure, it is possible to create any number of encoded blocks. LT decoding utilizes the fact that the encoded blocks are created using the EXCLUSIVE-OR of some data blocks. For example, when the encoded blocks $C_1 = I_3$, $C_2 = I_1 \oplus I_2 \oplus I_3$, $C_3 = I_1 \oplus I_3$ created from data blocks I_1, I_2, and I_3 are received, it is possible to first recover I_3 from C_1, then recover I_1 from $C_3 \oplus I_3$, and finally recover I_2 from $C_2 \oplus I_1 \oplus I_3$. If robust soliton distribution [14] is used for the probability distribution p_d, it is possible to limit the overhead required for decoding to approximately 5% [9].

In our research, we assume the use of LT codes. When several encoded blocks are created from k fixed-size data blocks of a file and are placed on peers in the network, the file can be restored by collecting any $(105/100)k$ of these blocks. (In our research, we refer to encoded blocks as "chunks," each chunk being the same size as a data block.) Furthermore, the use of rateless codes makes it possible to place additional chunks from the same file on different peers. If $(105/100)k$ chunks generated from the same file are collected, the file can be restored even if the chunks were created by different peers.

B. P2P systems using erasure codes

Previous research on structured P2P systems using erasure codes includes Rodrigues et al., [15] and Ribeiro et al. [16] Research on unstructured P2P systems that use erasure codes includes Cuenca-Acuna et al. [17] and Lin et al. [18] However, each of these studies focuses on the use of erasure codes for the objective of improving the file search success ratio; load distribution benefits resulting from the distributed placement of chunks created using erasure codes have not been confirmed.

In this paper, we focus not only on the file search success ratio but also on the time required to obtain a file, thus verifying the benefits of load distribution. We adopt a selection algorithm in which among the peers that have chunks of the target file, priority is given to the peers that can provide the chunks quickly. We compare download wait times of cases in which the files are placed on peers as is, cases in which the files are simply divided and placed on peers as chunks as in the case of BitTorrent [19], and cases in which the chunks placed on peers are created using erasure codes.

IV. Proposed technique

In our research, peers that wish to acquire a file use flooding to conduct searches, in which there may be multiple files conforming to the query. We propose a technique for collecting addresses of peers that may be candidates for chunk placement during the flooding. We also propose a technique for adding the addresses of peers on which other chunks created from the same file have been placed to chunks. We describe these techniques in the following subsections.

A. Adding address information to chunks

Before chunks are placed, they are augmented with peer address information that indicates where the other chunks that were created at the same time will be placed. Then, when one peer owning a chunk of a file is found, its address information can be used to identify other peers on which chunks were placed at the same time.

B. Chunk placement list

One possible technique for determining where to place file chunks after a file has been received is the path replication technique in which chunks are placed on peers in the search path. Another possible technique chooses among the peers adjacent to the search request peer. The path replication technique is not realistic because it concentrates the load on high-degree peers. In addition, it is inappropriate to use the concept of a search path in our technique because our technique allows to download chunks from peers identified through address information that has been added to chunks rather than from peers discovered by flooding.

Since the technique that places chunks on peers that are adjacent to the search request peer entails that any peer on which the chunk is placed can connect to another through one peer (two hops), adding address information to chunks is not effective. Furthermore, in cases where there are only a small number of adjacent peers, multiple chunks are placed on the same peer, and this prevents a fast spreading of the chunks.

In this research, we propose a technique in which a "chunk placement list" containing the addresses of peers that are chunk placement candidates is maintained by each peer, and the locations of placing chunks created from an obtained file are selected at random from this list. (If it is discovered that a chosen peer has been removed from the network, the peer is deleted from the list and another peer is selected.) The chunk placement list of a newly participating peer initially consists of the addresses of adjacent peers, and when the peer searches for a file, peers obtained from "flooding using flagged queries" are added to the chunk placement list.

The procedure for flooding using flagged queries is as follows:

1) The search request peer sends a flagged query to all adjacent peers.
2) A peer that receives a flagged query processes it in one of the following ways. (In addition, if a peer has a file or chunk that satisfies the query, it sends a response message to the search request peer as in normal flooding.)
   - If the TTL of the query is 0, the peer reports its own address to the search request peer and does not forward the query.
   - If the TTL is 1 or greater and the peer has previously received the query, the peer randomly chooses one peer address from its own chunk placement list, reports that address to the search request peer, and does not forward the query.
   - In all other cases, the peer forwards the query to all adjacent peers other than the sender of the query, and the search proceeds. At this point, a flagged query is sent to one randomly chosen adjacent peer and unflagged queries are sent to all other adjacent peers.
3) A peer that receives an unflagged query processes it in the same way as normal flooding.

The idea behind this procedure is that the chunk placement list maintained by each peer can store addresses of peers that are distant (on the overlay network) from that peer. As peers have large lists of candidate peer addresses, the distributed placement of chunks is made possible.

Considering the possibility that a peer may participate in the network for a long period of time, it may be necessary to limit the number of addresses that can be stored in the chunk placement lists (with old addresses deleted first in the case of overflow) or to set an expiry time. However, we do not impose such restrictions on the simulation carried out for our research.

V. Simulation

We test the performance of our proposed technique by conducting simulation experiments using a program written in C++. In this section, we first explain the model that we use to create the network. Then, we explain the simulation parameters, and finally, we present the results of the simulation.

A. Network model

It is known that real-world networks have the following three properties: scale-free (the degree distribution of peers follows the power law), small-world (SW; the average distance between peers is small), and clustered (the cluster coefficient is sufficiently large). In the Watts-Strogatz model [20], it is possible to create a network that is small-world and clustered, but the property of being scale-free is not met. The Barabási-Albert model [21] does not support clustering, and the growth and deactivation model [22] does not satisfy the small-world property. The SW growth and deactivation model [23] can create a network that satisfies all three properties. This model creates a network as follows.

1) Create a complete graph composed of m_0 (≥ 1) peers in activated states.
2) Add new peers in activated states with m ($\leq m_0$) links to the network. Each link connects the new peer and the peer selected by a) with a probability of μ $(0 < \mu < 1)$ and b) with a probability of $1-\mu$.
   a) Select a peer at random from all peers with the probabilities proportional to their degrees.
   b) Select a peer at uniform random from peers in activated states.

   If there is already a link that connects the new peer and the selected peer, the selection process is repeated.
3) Among the $m_0 + 1$ peers in activated states (including the newly added peer), select one at random with the probabilities inversely proportional to their degrees, and put it in a deactivated state.
4) Repeat steps 2 and 3 until the determined number of peers is reached.

We use this model to create a network for conducting our simulation experiments.

B. Simulation conditions

For our simulation experiments, we use the SW growth and deactivation model with $m_0 = 3$, $m = 3$, and $\mu = 0.4$, creating a network containing 5000 peers. There are 50 search keywords, and 10 unique files conform to one of these keywords. (No files match multiple keywords.) When the simulation is initialized, each of the 500 files is placed on a randomly selected peer.

The simulation progresses by clock ticks (each clock tick is called a "phase") and the following operations are carried out for each peer on the network in each phase.

1) Send a search request with a probability of 1.5% using the flagged-flooding search with the TTL set to 3. (One of the 50 search keywords is selected at random with equal probabilities.) Next, place files or chunks on other peers according to a chunk placement technique.
2) Remove the peer from the network with a probability of 1%.

If a peer is removed from the network in step 2, a new peer (a peer without any files or chunks and with an initialized chunk placement list) participates in the network. The new peer is connected to the adjacent peers of the removed peer, so the network topology during the experiment remains unchanged.

If a file (or chunks generated from a file) conforming to the search query is not found in step 1, the search fails. When conforming files are found, the search request peer selects a file to download from those files at random with equal probability. If the selected file is discovered in the form of a chunk, the chunks necessary for restoring the file are also collected using the address information in the discovered chunk, and the file is restored. Because it is possible that peers identified in the address information have been removed from the network or do not have the desired chunks any longer, a query is sent to each peer to check whether it has the desired chunks. At the same time, the predicted download completion time for downloading the chunk from that peer is calculated from the current file send status of that peer. It is then possible to prioritize the selection of a peer from which to download, choosing the peer that can provide the chunk in the shortest time. In our simulation, the transmission speed for each peer is fixed, so it is easy to calculate the predicted download completion time. In a real system, however, it will be necessary to have a method (such as a test transmission of a partial chunk) for roughly estimating the download completion time. In the case where a file is to be directly downloaded and the same file is found on multiple peers, one of the peers from which the file can be quickly downloaded is selected as is the case of a chunk.

Each peer uploads a file or chunk in response to a request from another peer. In order to observe the effects of the wide distribution of files and chunks on load distribution, we add a restriction that prevents simultaneous uploading to multiple peers. Thus, while a peer is uploading to a search request peer, if an upload request arrives from another peer, the latter peer is made to wait until the current upload is complete. (The length of this wait time is used as one of our evaluation indices.) Furthermore, uploading to a search request peer has priority over replica placement such that a peer carries out replica placement only when it is not currently uploading to a search request peer. If a peer receives an upload request from a search request peer while it is placing a replica, the upload request has priority and is processed, and after that is complete, the replica placement is restarted. The times taken for uploading one file and one chunk are counted as 20 phases and 1 phase, respectively (that is, the size of a chunk is 1/20 the size of a file). No restriction is placed on the download side, and so it is possible for a peer to simultaneously download from multiple peers.

C. Simulation results and discussion

This subsection presents the results of the simulation experiments and discusses the effectiveness of our proposed technique. The simulation is run 10 times under the same condition while reconstructing a network, and the averages for these simulations are plotted on a graph. The x-axis of the graph represents the phases, and the following three evaluation index values are plotted for each area, where an area represents 50 phases.

1) Search success ratio
   The "search success ratio" is defined as the ratio of the searches in which at least one file (or sufficient chunks for restoring the file) among the 10 files conforming to the keyword is found.
2) Recall ratio
   When a search is successful, the "recall ratio" is the ratio of discovered files (or sufficient chunks for restoring files) to the 10 files conforming to the keyword. For example, a recall ratio of 0.3 indicates that on average, three files conforming to the keyword are found.
3) Average queueing time
   The "average queueing time" is defined as the average wait time from when a file is requested until the download begins. (In the case of a file that must be reconstructed from chunks, the wait time is defined as the longest time for any of the chunks to be downloaded.)

1) Comparison of file placement techniques: First, we perform a simulation experiment (Experiment 1) to investigate the efficiency of a technique that uses chunk placement lists as compared with representative replica placement techniques for placing files such as owner replication and path replication. As shown below, we change the number of replicas and placement peers after obtaining the files, and we compare the performance.

- FileOR: Owner replication (the search request peer publishes the obtained file).
- FilePR: Path replication (In addition to owner replication, replicas are placed on all peers in the search path).
- File1AR: In addition to owner replication, one replica of the obtained file is placed on an adjacent peer of the search request peer.
- File2AR: In addition to owner replication, two replicas of the obtained file are placed on two adjacent peers of the search request peer.
- File1LR: In addition to owner replication, one replica of the obtained file is placed on a peer in the chunk placement list of the search request peer.
- File2LR: In addition to owner replication, two replicas of the obtained file are placed on two peers in the chunk placement list of the search request peer.

Figure 4 shows the search success ratios, Figure 5 shows the recall ratios, and Figure 6 shows the average queueing times for Experiment 1.

FileOR, which does not place replicas on peers other than the search request peer, has extremely low search success and recall ratios compared to the other techniques. There is no major difference between the other techniques. The search success and recall ratios tend to increase with the number of replica placements. It is known from a preliminary experiment that the number of replica placements (not including a file published by

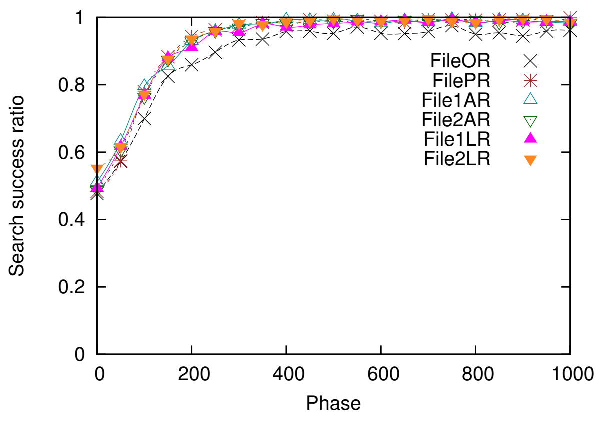


Figure 4. Search success ratio in Experiment 1.

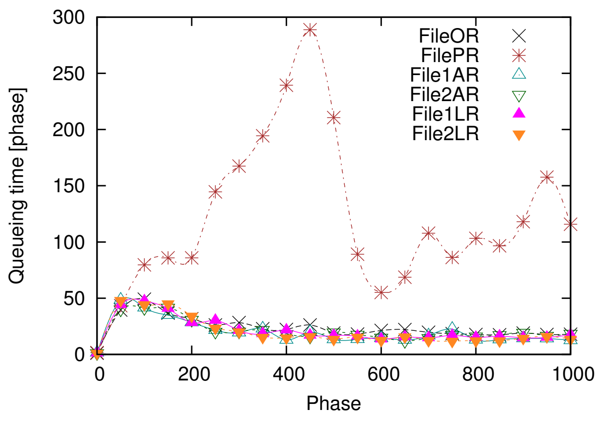


Figure 6. Average queueing time in Experiment 1.

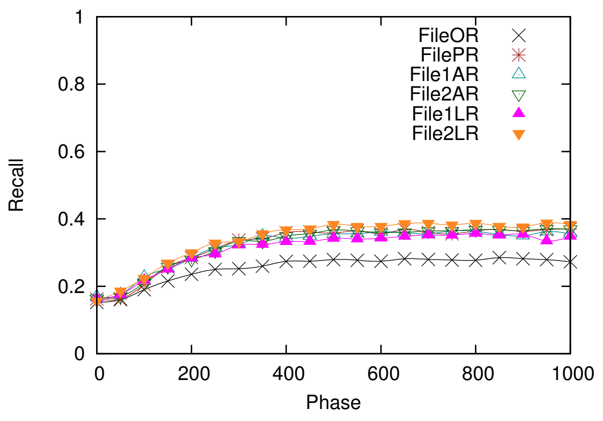


Figure 5. Recall ratio in Experiment 1.

the search request peer) in FilePR is between one and two.

There is a large difference between the average queueing time for FilePR and for other techniques. In FilePR, as replicas are placed on peers in the search path, many replicas are placed on high-degree peers that tend to be in the query's transmission path. Moreover, in searching with flooding, there is a high probability that a query arrives at a high-degree peer. For this reason, although a mechanism for selecting the optimal peer for the download is adopted, file upload requests are concentrated in high-degree peers and this increases the wait time.

In Experiment 1, no clear difference is observed between the techniques for placing on adjacent peers (File1AR, File2AR) and the techniques based on chunk placement lists (File1LR, File2LR). However, these techniques are demonstrated to be superior to owner replication (FileOR) and path replication (FilePR).

2) Comparison of chunk placement techniques: We execute a simulation experiment (Experiment 2) to verify the effectiveness of our proposed technique when placing chunks. To demonstrate the effectiveness of the proposed technique using rateless codes, the performance of the proposed technique is compared with the case in which file replicas are placed and the case in which a file is split into 20 equal-sized blocks numbered 1-20. Those blocks are also referred to as chunks. Address information is added to this type of chunks, similar to the technique using rateless codes. Once 20 chunks with different numbers are collected, the file can be restored, even if they were created by different peers. The technique using rateless codes has an overhead of 5% as described above, and the file can be restored when any 21 chunks created from the same file are collected.

For each of the three placement techniques, we consider the case where files and chunks are placed on adjacent peers and the case where files and chunks are placed on peers appearing in the chunk placement list. We perform a comparison of the following six techniques. (Since the size of one file is the same as the size of 20 chunks, these techniques are compared fairly in terms of the replica placement volumes.)

- File1AR: In addition to owner replication, one replica of the obtained file is placed on an adjacent peer of the search request peer.
- ChunkD20AR-20: In addition to owner replication, 20 chunks created by splitting the obtained file are placed on adjacent peers of the search request peer.
- ChunkR20AR-21: In addition to owner replication, 20 chunks created using rateless codes are placed on adjacent peers of the search request peer.
- File1LR: In addition to owner replication, one replica of the obtained file is placed on a peer in the chunk placement list of the search request peer.
- ChunkD20LR-20: In addition to owner replication, 20 chunks created by splitting the obtained file are placed on peers in the chunk placement list of the search request peer.
- ChunkR20LR-21: In addition to owner replication, 20 chunks created using rateless codes are placed on peers in the chunk placement list of the search request peer (proposed technique).

Figure 7 shows the search success ratios, Figure 8 shows the recall ratios, and Figure 9 shows the average queueing times for Experiment 2.

Our proposed technique (ChunkR20LR-21) produces

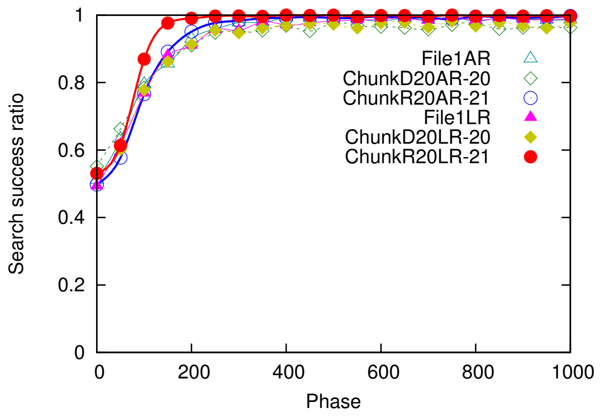


Figure 7. Search success ratio in Experiment 2.

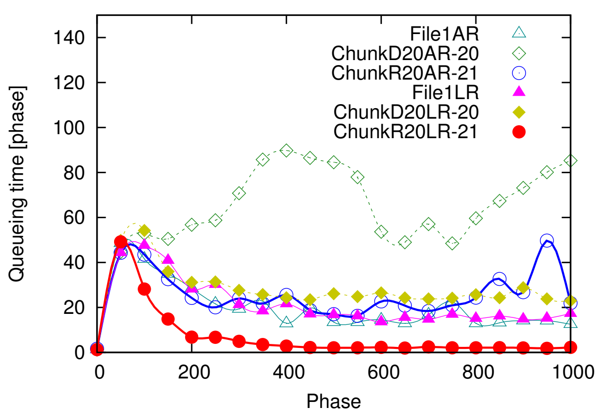


Figure 9. Average queueing time in Experiment 2.

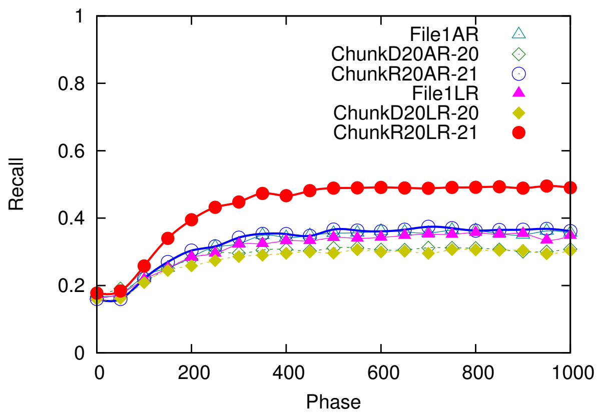


Figure 8. Recall ratio in Experiment 2.

the best search success and recall ratios, which suggests that this technique distributes the chunks in a highly efficient manner. Despite placing the same volume of replicas, the performance of the chunk placement technique for placing chunks created by splitting a file is inferior to that of the file placement technique. This is because when a chunk is absent owing to the removal of peers, it is not easy to find a chunk with the same number. In the case of the technique using rateless codes, it is possible to mitigate the decrease in search success and recall ratios because a file can be restored as long as chunks created from identical files are collected. Compared with ChunkR20AR-21, the proposed technique (ChunkR20LR-21) allows chunks to be distributed on more different peers, thus improving the search success and recall ratios.

ChunkR20LR-21 also has the best average queueing time performance. The wait time in early stages is reduced, and there is almost no wait after the distribution settles down. Since it is necessary to collect chunks with all numbers in technique ChunkD20LR-20, chunks must frequently be downloaded from a peer in which requests are concentrated. In contrast, this requirement can be avoided with a high degree of probability in ChunkR20LR-21. Furthermore, queueing time can be reduced in ChunkR20LR-21 because it is possible to distribute chunks in ChunkR20LR-21 more effectively than in ChunkR20AR-21, thus producing a larger pool of peers from which chunks can be downloaded. Of the two techniques for placing chunks created by splitting a file, ChunkD20LR-20 produces better results than ChunkD20AR-20, confirming that queueing time can be reduced by using a chunk placement list.

In this experiment, by enhancing rateless codes through the addition of address information to chunks and chunk placement lists to peers and by making it possible to restore a file if a certain number of chunks created from identical files are collected, our proposed technique demonstrated superior performance on all the performance indices. Furthermore, the proposed technique demonstrated superior performance in spite of the severe condition where the number of chunks required to restore a file exceeds the number of chunks placed at a time. This fact shows that the proposed technique, which utilizes rateless codes, is highly effective.

VI. Summary

In this paper, we proposed an effective technique for replica placement using rateless codes in an unstructured P2P network. In particular, we proposed a technique for collecting peer addresses that are candidates for chunk placement when searching with flooding and a technique that augments a chunk with the addresses of peers in which other chunks created from the same file have been placed. We ran a simulation considering the participation and removal of peers, demonstrating the superiority of our proposed technique with respect to the search success ratio, the recall ratio, and queueing time.

Future research topics include designing a way to reduce the communication load by eliminating unnecessary chunk placement in cases where chunks are already sufficiently distributed. It would also be beneficial to study a technique for effectively utilizing rateless codes in structured P2P networks as well.

References

[1] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, A survey and comparison of peer-to-peer overlay

network schemes, *IEEE Communications Surveys*, Vol.7, No.2, pp.72–93, 2005.

[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, *Proc. SIGCOMM '01*, pp.329–350, 2001.

[3] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, A scalable content-addressable network, *Proc. SIGCOMM '01*, pp.161–171, 2001.

[4] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, *Proc. Middleware 2001*, pp.329–350, 2001.

[5] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, Tapestry: An infrastructure for fault-tolerant wide-area location and routing, *U. C. Berkeley Technical Report UCB/CSD-01-1141*, 2000.

[6] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, Search and replication in unstructured peer-to-peer networks, *Proc. ACM ICS '02*, 2002.

[7] N. Bisnik and A. Abouzeid, Modeling and analysis of random walk search algorithms in P2P networks, *Proc. HOT-P2P 2005*, 2005.

[8] I.S. Reed and G. Solomon, Polynomial codes over certain finite fields, *Journal of the Society for Industrial and Applied Mathematics*, Vol.8, No.2, pp.300–304, 1960.

[9] D.J.C. MacKay, Fountain codes, *IEE Proceedings - Communications*, Vol.152, No.6, pp.1062–1068, 2005.

[10] C. Wu and B. Li, rStream: Resilient and optimal peer-to-peer streaming with rateless codes, *IEEE Transactions on Parallel and Distributed Systems*, Vol.19, No.1, pp.77–92, 2008.

[11] T. Schierl, S. Johansen, A. Perkis, and T. Wiegand, Rateless scalable video coding for overlay multisource streaming in MANETs, *Journal of Visual Communication and Image Representation*, Vol.19, No.8, pp.500–507, 2008.

[12] T. Gasiba, W. Xu, and T. Stockhammer, Enhanced system design for download and streaming services using Raptor codes, *European Transactions on Telecommunications*, Vol.20, No.2, pp.159–173, 2009.

[13] M. Luby, LT Codes, *Proceedings of the 43rd Symposium on Foundations of Computer Science*, 2002.

[14] D.J.C. MacKay, Information theory, inference, and learning algorithms, *Cambridge University Press*, 2003.

[15] R. Rodrigues and B. Liskov, High availability in DHTs: Erasure coding vs. replication, *Lecture Notes in Computer Science*, Vol.3640, pp.226–239, 2005.

[16] H.B. Ribeiro and E. Anceaume, Exploiting rateless coding in structured overlays to achieve data persistence, *Proc. the IEEE 24th International Conference on Advanced Information Networking and Application (AINA 2010)*, 2010.

[17] F.M. Cuenca-Acuna, R.P. Martin, and T.D. Nguyen, Autonomous replication for high availability in unstructured P2P systems, *Proc. the 22nd International Symposium on Reliable Distributed Systems (SRDS '03)*, 2003.

[18] W.K. Lin, C. Ye, and D.M. Chiu, Decentralized replication algorithms for improving file availability in P2P networks, *Proc. the 15th IEEE International Workshop on Quality of Service (IWQoS 2007)*, 2007.

[19] B. Cohen, Incentives build robustness in BitTorrent, *Proc. the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[20] D.J. Watts and S.H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, Vol.393, No.6684, pp.440–442, 1998.

[21] A.-L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, Vol.286, No.5439, pp.509–512, 1999.

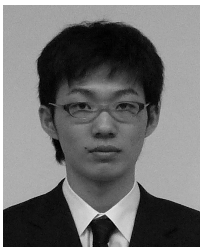
[22] K. Klemm and V.M. Eguíluz, Highly clustered scale-free networks, *Physical Review E*, Vol.65, No.3, pp.036123-1–036123-5, 2002.

[23] K. Klemm and V.M. Eguíluz, Growing scale-free networks with small-world behavior, *Physical Review E*, Vol.65, No.5, pp.057102-1–057102-4, 2002.

Keiichi Endo was born in Osaka, Japan in 1980. He received his B.S. degree in Engineering, M.S. and Ph.D. in Informatics from Kyoto University, Japan, in 2003, 2005, and 2008, respectively.

He is currently a Senior Assistant Professor at the Graduate School of Science and Engineering, Ehime University, Japan. His current research interests are in the area of peer-to-peer and wireless networks.

Dr. Endo is a member of IPSJ, IEICE, and JSIAM.



Ryosuke Hamabe was born in Kagawa, Japan in 1988. He received his B.S. degree in Engineering from Ehime University, Japan in 2010.

He is currently a graduate student at the Graduate School of Science and Engineering, Ehime University, Japan. His current research interests are in the area of peer-to-peer networks.

Mr. Hamabe is a student member of IEICE.

Dai Okano had B.Eng. and M.Eng. in Applied Physics, Dr.Eng. in Information Science and Technology from the University of Tokyo.

He is currently an Associate Professor at the Graduate School of Science and Engineering, Ehime University, Japan. His research interests are in Mathematics, Computations, and their applications.

Dr. Okano is a member of JSIAM, IPSJ, and SIAM. He received the 40th Anniversary Best Paper Award in 2000 from IPSJ.

Kaname Amano completed the B.Eng. and the M.Eng. in 1971 and 1974 from Kyoto University, Japan; then the Dr.Eng. in 1978 from Hokkaido University, Japan.

He is currently a Professor at the Graduate School of Science and Engineering, Ehime University, Japan. His current research area includes computational mathematics, cognitive psychology and computer science.

Prof. Amano is a member of MSJ, JSIAM, IPSJ, JPA, SIAM, etc. He received the 30th Anniversary Best Paper Award in 1990 and 40th in 2000 from IPSJ, and the Best Paper Award in 1996 from JSIAM.