IEICE TRANS. INF. & SYST., VOL.E94–D, NO.10 OCTOBER 2011

PAPER

# Query-Trail-Mediated Cooperative Behaviors of Peers in Unstructured P2P File Sharing Networks

**Kei OHNISHI**[†a)], **Hiroshi YAMAMOTO**[††b)], **Masato UCHIDA**[†††c)], ***Members***, ***and*** **Yuji OIE**[†d)], ***Fellow***

**SUMMARY** We propose two types of autonomic and distributed cooperative behaviors of peers for peer-to-peer (P2P) file-sharing networks. Cooperative behaviors of peers are mediated by query trails, and allows the exploration of better trade-off points between file search and storage load balancing performance. Query trails represent previous successful search paths and indicate which peers contributed to previous file searches and were at the same time exposed to the storage load. The first type of cooperative behavior is to determine the locations of replicas of files through the medium of query trails. Placement of replicas of files on strong query trails contributes to improvement of search performance, but a heavy load is generated due to writing files in storage to peers on the strong query trails. Therefore, we attempt to achieve storage load balancing between peers, while avoiding significant degradation of the search performance by creating replicas of files in peers adjacent to peers on strong query trails. The second type of cooperative behavior is to determine whether peers provide requested files through the medium of query trails. Provision of files by peers holding requested files on strong query trails contributes to better search performance, but such provision of files generates a heavy load for reading files from storage to peers on the strong query trails. Therefore, we attempt to achieve storage load balancing while making only small sacrifices in search performance by having peers on strong query trails refuse to provide files. Simulation results show that the first type of cooperative behavior provides equal or improved ability to explore trade-off points between storage load balancing and search performance in a static and nearly homogeneous P2P environment, without the need for fine tuning parameter values, compared to replication methods that require fine tuning of their parameters values. In addition, the combination of the second type and the first type of cooperative behavior yields better storage load balancing performance with little degradation of search performance. Moreover, even in a dynamic and heterogeneous P2P environment, the two types of cooperative behaviors yield good ability to explore trade-off points between storage load balancing and search performance.

***key words:*** *P2P file sharing, storage load balancing, file replication, cooperative behaviors, query-trail*

## 1. Introduction

Peer-to-peer (P2P) network models have recently attracted a great deal of attention. One of the applications of P2P networks that has attracted interest is a distributed storage system for file sharing. A distributed storage system for file sharing provides a large amount of storage by accumulating the unused storage of peers, which enables large amounts of data to be stored and shared without the need for a costly file server. According to [1], there are several forms in P2P networks for file sharing. We can roughly classify P2P networks for file sharing into two types. One type is structured P2P networks that have a mechanism to manage file locations in a network. Representatives of structured P2P networks are those using a distributed hash table (DHT) [2]–[5]. The other type is unstructured P2P networks that do not have such a mechanism, in which peers freely link to each other.

At present, pure unstructured P2P file sharing networks that are being actively used in the real world are decreasing in number. For instance, LimeWire [6] built upon the Gnutella network, which is a representative unstructured P2P file sharing network, was previously a pure unstructured P2P file sharing network, but the present LimeWire includes a DHT-based structured P2P network to achieve reliable search for particular files. However, the core component of LimeWire is still an unstructured P2P network, and due to the flexibility of an unstructured P2P network, it seems to be able to hold millions of active users at any given moment. Therefore, studies on pure unstructured P2P networks, especially large-scale ones, would be necessary.

To make such unstructured P2P networks more practical, they have been investigated from two perspectives. One is to enhance search performance and the other is to balance storage load among peers. To enhance search performance in unstructured P2P networks, query forwarding methods and file replication methods have been investigated. The basic strategy to enhance search performance in unstructured P2P networks is that a file replication method places more files at easy reachable peers by a given query forwarding method or that a query forwarding method forwards queries to peers holding more files. Such methods are essential for efficient file sharing because no mechanism is available for determining file locations.

However, enhancing search performance means that particular peers are more frequently accessed, thus causing a load bias to the particular peers. This problem is actually similar to the problem of one point failure in client-server network services. Thus, there is a trade-off relationship between search performance and storage load balancing in unstructured P2P networks, and therefore, unstructured P2P networks should include not only strategies to

Manuscript received March 11, 2011.
Manuscript revised May 31, 2011.

[†]The authors are with Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.
[††]The author is with Department of Electrical Engineering, Nagaoka University of Technology, Nagaoka-shi, 940–2188 Japan.
[†††]The author is with Network Design Research Center, Kyushu Institute of Technology, Chiyoda-ku, Tokyo, 100–0011 Japan.
a) E-mail: ohnishi@cse.kyutech.ac.jp
b) E-mail: hiroyama@nagaokaut.ac.jp
c) E-mail: m.uchida@ndrc.kyutech.ac.jp
d) E-mail: oie@cse.kyutech.ac.jp
DOI: 10.1587/transinf.E94.D.1966

enhance search performance but strategies to balance load among peers as well to reliably sustain themselves.

In the present study, we focus on unstructured P2P networks. The goal of the present study is to explore better trade-off points between storage load balancing and search performance in unstructured P2P file sharing networks. We therefore propose two types of autonomic and distributed cooperative behaviors of peers. In addition, we examine the ability of the two types of cooperative behavior of peers to explore trade-off points between storage load balancing and search performance through simulation experiments. Cooperative behaviors of peers are mediated by query trails. Query trails represent information of previous successful search paths and demonstrate which peers contributed to previous file searches and were at the same time exposed to the storage load. The strength of query-trails basically stands for access frequency to storage of peers.

We consider two types of storage load in the present paper. The first type of storage load is caused by writing a file in a peer. The meaning of writing a file in a peer is that a peer receiving a request to create a replica of a file stores the file in its storage. The second type of storage load is caused by reading a file from a peer. The meaning of reading a file from a peer is that a peer holding a requested file retrieves the requested file from its storage to forward the file to a peer making the request. The two types of cooperative behaviors of peers are proposed to balance these two types of storage load.

The first type of cooperative behavior of peers basically aims at balancing load among peers caused by writing files in the peers. This cooperative behavior of peers is actually equivalent to a file replication method and determines the locations of replicas of files through the medium of query trails. However, this first type of cooperative behavior of peers cannot handle balancing load among peers caused by reading files from the peers. The second type of cooperative behavior of peers basically aims at balancing load among peers caused by reading files from the peers. The second type is to determine whether peers refuse to provide files through the medium of query trails.

The present paper is organized as follows. Section 2 describes related research. Section 3 explains the proposed replication method, which is the first type of cooperative behavior of peers mediated by query-trails mentioned above. The proposed method is experimentally evaluated assuming a static and nearly homogeneous P2P environment in Sect. 4. In Sect. 5, we propose a method for balancing the load of reading files from storage, which is the second type of cooperative behavior of peers mediated by query-trails. In addition, the proposed method is combined with the replication method proposed in Sect. 3, and the combined method is examined experimentally assuming a static and nearly homogeneous P2P environment. In Sect. 6, the combined method is examined experimentally assuming a dynamic and heterogeneous P2P environment in contrast to the environment assumed in Sect. 5. Our conclusions are presented in Sect. 7.

## 2. Related Work

In this paper we propose a replication method for unstructured P2P networks and a method for determining a peer that provides a requested file in unstructured P2P networks. These proposed methods are intended to explore better trade-off points between search and storage load balancing performances. So, we here briefly describe previous studies of replication methods for unstructured P2P networks, methods for determining peers that provide services in unstructured P2P networks, load balancing methods for unstructured P2P networks, and methods for exploring some performance trade-off in unstructured P2P networks.

Replication methods are used in unstructured P2P networks for enhancing search efficiency and reliability [7]–[11]. Since searching in unstructured P2P networks is blind, replicas of files distributed over networks contribute to quick and reliable file searching. In [11], several previous replication methods are reviewed from two aspects, which are selection of files for replication and selection of sites for hosting new replicas, and a new replication method is also proposed. The replication method proposed in [11], Q-replication, selects files for replication based on calculation of popularity of the files in each peer. Also, it selects peers for hosting new replicas from among peers reachable from a peer of focus by random-walk with limited hop counts, in which the ability and availability of the candidate peers for hosting new replicas are considered to be selected. These ways are different from those of our replication method, but how to select peers for hosting new replicas is methodologically similar to ours in terms of selecting the candidate peers by random walk. In this way, peers of high degree are likely to be selected.

Replication methods are also used in unstructured P2P networks for access load balancing [12]. In [12], the load on a peer is defined as the number of bytes transfered from the peer as a response to queries originated from other nodes. The load index is the Fairness Index, which is basically the same as the Balance Index [13] presented and used in Sect. 6. The replication method presented in [12] relies only on local communication between peers to achieve load balancing. The peers locally exchange their load states and examine if they are overloaded compared to the neighboring peers using their load states and a threshold. If a peer is regarded as an overloaded peer, then the peer replicates the most frequently queried item among its own items on a neighboring peer that the most frequently or the second frequently forwards the query for the item to the peer, if the replication is expected to better achieve load balancing. Our replication method presented in this paper also relies only on local communication among peers to achieve load balancing and makes the peers judge if they are allowed to create replicas of items on other peers based on their load states, which correspond to query trails in the present paper. However, our replication method tries to create replicas of items just when a successful search path for a certain query

is obtained, and candidate peers on which the replicas will be created are those on the successful search path and their neighboring peers. In addition, replicas of items are likely to create on less loaded peers, rather that peers that frequently forwarded queries to overloaded peers.

Our previous paper [14] focused on replication methods in unstructured P2P networks for file sharing. Our objective was not only to achieve good search performance but also to achieve storage load balancing. In particular, we proposed three different replication methods in our previous paper: Path Random Replication, Path Adaptive Replication, and Path Adaptive Replication with Priority Level. These methods are all probabilistic replication methods that make replicas of files in peers on the present successful search path. Unlike our previous replication methods presented in [14], the replication method presented in this paper considers the relationship between the states of storage loads of peers adjacent to the peer of focus in order to achieve storage load balancing among peers.

There are several methods for selecting peers that provide services based on given metrics. Peers providing services like files in a file sharing network are selected in order for the network to reduce cross-ISP traffic in [15] and to minimize downloading time for files in [16]. In this paper, we propose a method for selecting peers that provide requested files in order for the network to balance the load of reading files from the storage among peers.

As approaches to load balancing in unstructured P2P networks, dynamic topology reconfiguration schemes [17], [18] have been proposed. The topology reconfiguration schemes basically dynamically move logical incoming links of overloaded peers to underloaded peers. The method presented in [17] is meant to balance load caused by queries among peers by the topology reconfiguration, in which virtual servers mediate between overloaded and underloaded peers.

There are several studies that seek to explore certain performance tradeoff in unstructured P2P networks [19], [20]. For example, in [19], the tradeoff between overall system performance and fairness to high bandwidth users in BitTorrent-like unstructured P2P systems is considered. In the present paper, we seek to explore the performance tradeoff between access load balancing and search in unstructured P2P networks.

Our previous paper relevant to the present paper is [21]. In [21], the first type of cooperative behavior of peers mentioned in Sect. 1 was proposed and evaluated through simulations. In the present paper, we will newly propose another type of storage load balancing method, which balances load caused by reading a file from a peer, and also, explain the two types of storage load balancing methods proposed in [21] and the present paper by a concept of *query-trail-mediated cooperative behaviors of peers*. In addition, we will evaluate the proposed cooperative behaviors of peers in dynamic P2P environments through simulations.

## 3. Replication Method

In this section, we present the first type of cooperative behaviors of peers, which is a new replication method for unstructured P2P networks for file sharing. The proposed replication method determines where to place replicas of files through the medium of query trails. The new method is meant to balance the storage load among peers as well as to limit the increase in the number of hops needed to find requested files.

### 3.1 Concept

In the present study, we use a random-walk-based query forwarding method. In fact, under the use of random-walk-based query forwarding methods, differences in degree of peers could induce differences in the frequency of query arrival to the peers. Concretely, when a random walker moves around in an arbitrary network, the probability with which the random walker arrives at a node is proportional to the degree of the node [22]. In such a case, if P2P networks select only peers on the present successful search path as peers in which a replica of a requested file is created, it is likely that the rate at which the number of files in peers of high degree increases is higher than that in peers of low degree. In addition, it is likely that queries made by peers of low degree are propagated by way of peers of high degree. That is, peers of low degree depend strongly on peers of high degree with respect to file searching. Therefore, in order to achieve load balancing in P2P networks, peers of low degree should bear part of the overall load that peers of high degree require to achieve load balancing in P2P networks.

It is, however, true that peers of high degree have to hold several replicas of files in order to handle the increases in the number of hops. To overcome the trade-off between storage load balancing and search performance, we will place several replicas of files in peers adjacent to a peer of high degree so that the peers adjacent to the peer of high degree can provide with certainty requested files that the peer of high degree does not provide. This method would not increase the number of hops greatly. However, if several replicas of files are placed in peers of high degree adjacent to a peer of high degree, the situation would be the same as in Path Replication and Path Random Replication, in which replicas of files are placed in peers on the present search path. Therefore, the key to simultaneous achievement of load balancing and reduction of the number of hops is the method by which replicas of files are allocated around peers of high degree. That is, replicas of files should be placed in peers with less load that are adjacent to peers of high degree.

When using a random-walk-based query forwarding method and a file replication method that create replicas of files in peers on the present successful search path, degree of peers can be thought to be correlated with the storage load of peers, and therefore, can be used as values to estimate the storage load of peers. On the other hand, in a different sit-

uation from this, it is not sure that degree of peers works as values to estimate the storage load of peers. Therefore, we should prepare values to estimate the storage load of peers in a variety of situations including the situation above.

In the present study, we use query trails as values to estimate the storage load of peers. The query trails represent information on the previous successful search paths that queries passed through. However, since it is impossible to leave query trails on network links, each peer memorizes which directed links from the peer to its neighboring peers queries passed through. In other words, each peer memorizes which of its neighboring peers it forwarded queries to. We regard peers through which queries frequently passed and to which queries were frequently forwarded by their neighboring peers as peers of high load.

We have two reasons for choosing the strength of query trails as values that represent the storage load of peers from among various possible values to estimate the storage load of peers. One reason is technical. The strength of query trails can reveal which peers were subjected to storage loads by previous searches, no matter what query forwarding methods are used. The other reason is conceptual. The replication method mediated by query trails is analogous in some sense to an ant colony mediated by pheromone trails. An ant colony has a function whereby a group of ants finds the shortest path from an anthill to food, so that the analogy of an ant colony has been used for methods by which to search for the shortest path [23], [24] and path with the lowest cost on the Internet [25]. Here, we focus on the feedback of actions of elements with a small function, such as ants or peers, to the entire function, rather than on a function to find the path with the lowest cost. Whereas the entire function whereby a swarm of ants forms through the medium of pheromone trails is to find the shortest path from an ant hill to a food source, the entire function whereby a swarm of peers forms through the medium of query trails is the creation of file replications capable of storage load balancing and practical file search performance.

### 3.2 Method

Next, we will explain in detail the proposed replication method. The basic feature of the proposed replication method is to determine a peer in which a replica of a requested file is placed by using the history of the previous successful searches, which is equivalent to a query trail. The history is managed not by particular peers but by each peer. If particular peers manage the history as in a client-server network, their disappearance from a network would damage the overall functionality of the network. On the other hand, by allowing each peer to manage only its own search experience, the disappearance of peers would not greatly affect the overall functionality of a network.

The proposed replication method repeats the three steps described below.

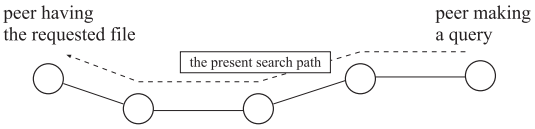



**Fig. 1** An example of obtaining a successful search path. The circles represent peers, and the lines between circles represent links.

(1) Obtaining a successful search path
A given query forwarding method first finds a file requested by some peer and then obtains the path from the peer that makes a query to a peer having a requested file as the present successful search path (see Fig. 1). If a given query forwarding method finds several successful search paths at the same time, the shortest path among all of the successful search paths is regarded as the present successful search path.

(2) Selecting peers for placing replicas of files
Peers on the present successful search path and their neighboring peers have possibility of being selected as peers in which a replica of a requested file is created probabilistically. Two measures of query trails strength are used for the selection of peers.
Suppose that a certain peer on the present successful search path, $q$, is linked to $Q$ peers. First, the number of times that peer $q$ was on the previous successful search paths, $O_q$, and the average number of times that its $Q$ neighboring peers were on them, $\overline{O}_Q$, are obtained, as given by Eq. (1):

$$\overline{O}_Q = \frac{\sum_{i=1}^{Q} O_i}{Q}, \tag{1}$$

where $O_i$ $(i = 1, 2, \ldots, Q)$ is the number of times that the $i$-th peer in the $Q$ neighboring peers was on the previous successful search paths. This is one of the two measures of the strength of the query trails.
Next, according to the comparative results between $O_q$ and $\overline{O}_Q$, which fall into either of the two cases described below, the replication method selects a peer in which a replica of a requested file is created probabilistically and then actually makes the replica in the selected peer with fixed probability. This fixed probability is a parameter of the proposed method and is hereinafter referred to as the replication probability (**RP**). As shown below, $\overline{O}_Q$, as represented by Eq. (1), adds two different types of replication strategy into each of the local networks that comprise the entire network.

(a) Case 1: $O_q \leq \overline{O}_Q$
Peer $q$ is simply selected as a peer in which a replica of a requested file is created with a given replication probability. This means that if the number of times that the peer of interest was on the previous successful search paths is smaller than the average number of its neighboring peers, the peer of interest is explicitly prohibited from selecting a peer in which a replica of a requested

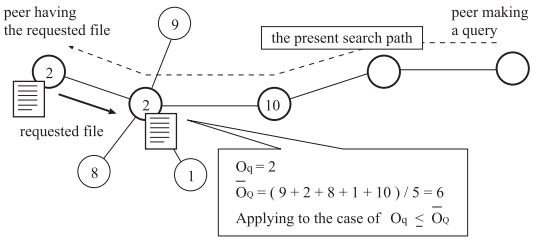



**Fig. 2** An example of selecting a peer in the case of $O_q \leq \overline{O}_Q$. The circles represent peers, and the lines between circles represent links. The numbers in the circles indicate the number of times that the peers were on previous successful search paths.

file is created probabilistically (referred to as *a peer for replication* hereinafter) from among its neighboring peers. An example of this procedure is shown in Fig. 2.

(b) Case 2: $O_q > \overline{O}_Q$

The replication method selects one peer from among the peers adjacent to peer $q$, where the *points* explained in Step (3) below are used.

The points are represented by positive integers. The points are the second measure of the strength of the query trails. Suppose that peer $q$ previously assigned $P_i$ $(i = 1, 2, \ldots, Q)$ points to each peer adjacent to peer $q$. The probability with which the $i$-th peer among the $Q$ peers is selected as a peer for replication is given by Eq. (2):

$$S_i = \frac{\frac{1}{1+P_i}}{\sum_{j=1}^{Q} \frac{1}{1+P_j}} \tag{2}$$

This equation indicates that peers to which more comparatively points were assigned in the past can prevent themselves from being selected as peers for replication. The replica is made in the selected peer according to the above equation with the given replication probability. An example of this procedure is shown in Fig. 3.

In the procedure described above, if a selected peer already has a requested file, then a replica of the requested file is not made in the selected peer. In addition, the procedure described above is applied to all of the peers on the present successful search path sequentially in a direction from the peer having the requested file to that making the query.

(3) Updating data of peers

Let the direction to a peer having a requested file on a successful search path be the upper direction on the successful search path. Each peer on the present successful search path increases by one the number of times it forwarded a query to the peer in the upper direction on the path. The number of times a query is forwarded in this manner is hereinafter referred to as *points*, and the phrase "assigning a point to a peer" is also used herein (see Fig. 4).

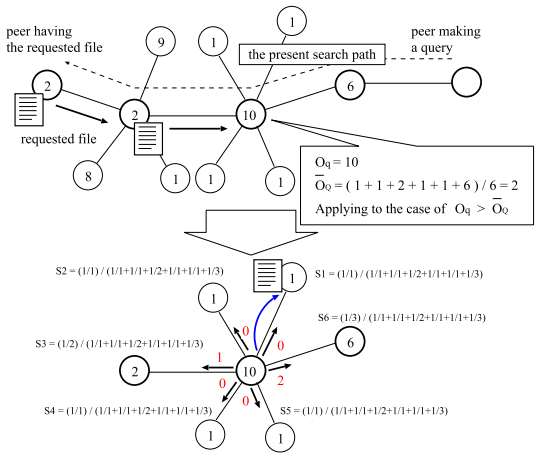



**Fig. 3** An example of selecting a peer in the case of $O_q > \overline{O}_Q$. The circles represent peers, and the lines between circles represent links. The numbers in the circles indicate the number of times that the peers were on previous successful search paths. The numbers next to the center peer that had 10 previous successful search paths indicate the points that the peer assigned to neighboring peers denoted by the arrows. In this example, each of the six peers had a chance to be selected with probability $S_i$ $(i = 1, 2, \cdots, 6)$ and the first peer with probability $S_1$ has happened to be selected.

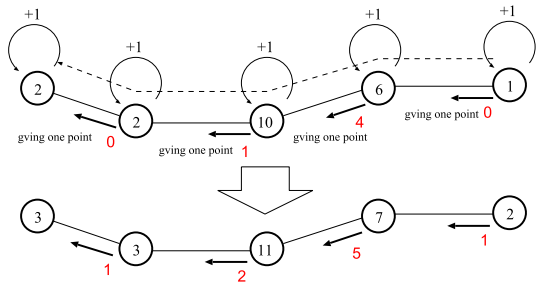



**Fig. 4** Updating the *points* of the peers on the present successful search path. The numbers in the circles indicate the number of times that the peers were on previous successful search paths. The numbers next to each peer indicate the points that the peer assigned to neighboring peers denoted by the arrows.

This *point* is one of the two measures of the strength of the query trails, where the directions of query trails are taken into account. The other measure of the strength of the query trails is the number of times that peers were on the previous successful search paths, where the directions of query trails are not taken into account. This is actually $O_i$ mentioned in the explanation of Step (2).

As for point assignment records, peers that assigned points to other peers record to which peers they assigned points, along with the total number of points they assigned to the peers in the past. The points mentioned here are used in Step (2)–(b) above for selecting peers for replication. After assigning points, all of the peers on the present successful search path, including peers that made a query and supplied a requested file, increase the number of times that they were on the previous successful search paths by one (see Fig. 4).

The initial points and the number of times of being on search successful paths are zero for all of the peers composing the network.

The proposed replication method is called the Query-trail-based Replication, referred to hereinafter as **QR**. In Sects. 4 and 5, none of the peers is allowed to reduce the number of times that it was on previous successful search paths and points assigned to its neighboring peers. However, in practice, the storage load of peers (access frequencies) over a short span must be accurately estimated, for example, by reducing the number of times that the peers were on previous successful search paths and the points by a certain amount at a certain interval.

Finally, to implement QR in a P2P network, every peer has to gather the number of times that a peer was on the previous successful search paths from its neighboring peers whenever it determines which neighboring peer to create a replica of a file in. This information gathering causes additional processing and network traffic to the network. However, the amount of the information exchanged among the peers to implement QR is negligibly small compared to the size of a file shared among the peers, because the exchanged information is just an integer. Therefore, the cost of the information gathering mentioned here will not be discussed hereinafter.

## 4. Experiments

In the previous section, we proposed the replication method, QR. Next, we will experimentally evaluate QR with respect to both storage load balancing and search performance.

### 4.1 Simulation Model

The configurations of the P2P network simulation model are as follows. The total number of peers present in the network is 10,000 and the total number of links between peers is 20,000. The distribution of the degrees of peers in the network is shown in Fig. 5. The topology of the P2P network in the simulation model used is generated by the algorithm described in [26]. The topology follows a power law [27], [28] with respect to the distribution of degree. Since the Gnutella network is a representative unstructured P2P network that follows an approximate power law with respect to the distribution of the degree of peers [29], it is valid to use network topology following a power law for the investigation of unstructured P2P networks. So, we will evaluate the proposed methods assuming a static power-law network topology. However, it is shown in [30] that the network topology of the recent Gnutella does not follow a power-law, if it is build within a short period of time. That would be because the peers frequently leave and join the network. Therefore, we will consider a dynamic change in a P2P network topology which is based on [30] and evaluate the proposed methods under that situation in Sect. 6.

The maximum storage capacity of every peer is 20.

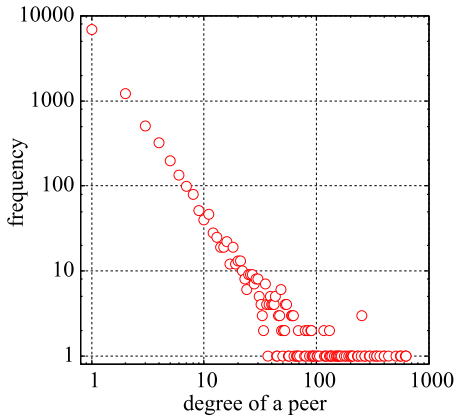



**Fig. 5** Distribution of the degrees of peers in the network used herein. This distribution follows a power law.

One file consumes one unit of storage, so that the maximum number of files that a peer can hold is 20. When the storage of a peer is full and a request for a replica of a requested file arrives at the peer, the oldest file is replaced by the requested file (FIFO replacement discipline). How to displace the files in storage of a peer by new one when the storage is full would somewhat influence the storage load balancing and the search performance considered here. However, we use the same way of file displacement for several replication methods compared here and will not discuss the effects of file displacement ways.

The query forwarding method used in the present paper is a 16-walker random walk [9]. The 16-walker random walk literally uses 16 walkers with a query that randomly walks around peers from the peer making a query for a requested file. Even if the 16 walkers start from a peer with degree less than 16, each of the 16 walkers just randomly selects one of neighboring peers of the peer as a next-hop peer. The Time To Live (TTL) for every walker is 100 hops. In addition, a walker may revisit the same peer more than once. If one or more of the 16 walkers finds a requested file, then the shortest path among the successful search paths that they obtained is selected as the present successful search path and the search is finished.

In fact, flooding as a query forwarding method is easy to implement in unstructured P2P networks and often used for search in them. Flooding forwards a query arriving at a peer to all of neighboring peers of the peer, and provides reliable search if TTL is set to be large. However, when using flooding with a large value of TTL, the number of query messages forwarded over a network becomes vast. It is shown in [9] that combination of $k$-walker random walk as a query forwarding method and a replication method based on pre-acquired access frequency of files provides as reliable search as flooding with smaller number of query messages. We do not use the same replication method presented in [9], but expect to achieve reliable search by the combination of 16-walker random walk and each of the replication methods used in the present paper with smaller number of query messages.

In one run of the simulation model, the search for a requested file is repeated 50,000 times in a one-by-one manner, and a peer that makes a query is randomly chosen. Furthermore, a file requested by each peer is randomly determined. This means that all types of files in the network have an equal popularity.

The total number of file types in the network is 110. However, only 100 file types exist in the initial state of the network, and 10 files are allocated over the network with respect to each file type. The initial distribution of files is random, but identical, each time the simulation model is run. Next, immediately after the 10,000 searches have finished, another 10 file types are allocated over the network, where the number of files of each type is 10. Similar to the initial distribution, the distribution of the additional files is random, but identical, each time the simulation model is run.

One reason for increasing the number of types of files distributed over the network in the P2P simulation model as mentioned above is that it is a general event in real P2P file sharing networks. Another reason for this is to see how load of peers that was caused by search for the initially distributed 100 types of files influences the search performance for the lately distributed files. If we determine places of creating replicas of files by regarding storage utilization ratios as load of peers, creating replicas of files in peers whose storage is fully occupied by replicas of files is not promoted because those peers are considered to be highly loaded. If we define storage utilization ratios as load of peers in the present paper, peers of high degree that play an important role in quick and reliable file search first get their storage to be full occupation by replicas of the initially distributed files, and then they cannot hold replicas of the lately distributed files in their storage. Meanwhile, we define access frequency to peers (the strength of query-trails) as load of peers in the present paper and think it possible to explore the performance trade-off by updating the access frequency at short intervals in the situation of dynamic increasing and decreasing the number of types of files in the network. However, since the access frequency to peers are not statistically changed with the time in Sects. 4 and 5, it is not updated at certain intervals.

### 4.2 Evaluation Criteria

We will basically use the evaluation criteria described in our previous study [14]. In [14], the storage load was defined using not the distribution of storage usage ratios among peers, but rather the number of storage accesses.

Let $d$, $n_w$, and $n_r$ be the degree of a peer, the number of times a file is written in the peer, and the number of times a file is read from the peer, respectively. First, we plot three types of points, $(d, n_w + n_r)$, $(d, n_w)$, and $(d, n_r)$, with respect to all of the peers in the network and then fit a straight line to each of the three types of sets of points using the least squares method. The slope of a line fitted to a set of points of $(d, n_w + n_r)$ represents the evaluation criterion of storage load balancing (referred to hereinafter as **SL**). The smaller the slope, the greater the ability of the replication method in storage load balancing. Similar to SL, the slopes of two lines fitted with sets of points of $(d, n_w)$ and $(d, n_r)$ represent the evaluation criteria of writing and reading load balancing, respectively (referred to hereinafter as **WL** and **RL**).

The evaluation criterion for the search performance is based on the average number of hops needed to find requested files during some period. The average number of hops during some period is calculated on both the initially allocated files and the additionally allocated files. As for the initially allocated files, the number of hops from the 10,001th to 30,000th searches is taken into account for the calculation of the average number of hops (referred to hereinafter as **HI**). The reason for considering these 20,000 searches for the calculation of the average number of hops is as follows. Since the number of files of each type is quite small just after the files start being shared, the number of hops varies widely at the beginning of sharing those files. So, we consider the number of hops observed in a more stable situation, that is, from the 10,001th to 30,000th searches. As for the additionally allocated files, the number of hops from the 20,001th to 40,000th searches is taken into account for the calculation of the average number of hops (referred to hereinafter as **HA**). The reason for considering these 20,000 searches is the same as the case of the initially allocated files.

In addition to storage load and the average number of hops, we observe the total number of times a file is written for all of the peers in the network during all of the searches (referred to hereinafter as **NW**) and the total number of files in the network after all of the searches are finished (referred to hereinafter as **NF**).

All of the evaluation criteria are shown in Table 1.

### 4.3 Methods for Comparison

We use two replication methods for comparison to QR.

One is Path Random Replication (referred to hereinafter as **PRR**), which makes a replica of a requested file only in peers on the present successful search path with fixed probability. If the probability is set to 100%, then Path Random Replication is equivalent to Path Replication, which makes a replica of a requested file in all of the peers on the present successful search path. QR attempts to achieve better storage load balancing between peers than a replication method that places replicas of files on strong query trails, while not significantly degrading the search perfor-

**Table 1** Evaluation indexes.

| | |
|---|---|
| SL | Storage load balancing |
| WL | Writing load balancing |
| RL | Reading load balancing |
| HI | The average number of hops for initially distributed files |
| HA | The average number of hops for additionally distributed files |
| NW | The total number of times of writing files |
| NF | The total number of files |

mance compared to a replication method that places replicas of files on strong query trails, by creating replicas of files in peers adjacent to peers on strong query trails. Therefore, the most relevant existing method for comparison is exactly PRR, which creates replicas of files on strong query trails. In addition, it is expected that the number of times that replicas of the requested files are created is approximately the same for PRR and QR with the same replication probability. If this is true, we can say that the comparison of PRR and QR with the same replication probability is fair.

The other replication method makes a replica of a requested file in peers on the present successful search path with a probability that is inversely proportional to the degree of the peers, $rp$, as represented by $rp = \frac{C}{d} \in [0, 1]$, where $rp$ is equal to one if $C/d > 1$, $C$ is a constant, and $d$ is the degree of a peer. This method is called Replication with Probability Inversely proportional to the Degree of a peer, or **RPID**. We proposed PRID for comparison with QR. In RPID, the product of the probability of query arrival at a peer and the replication probability is approximately the same in every peer under the assumption that the number of times that a query arrives at a peer is proportional to the degree of the peer.

Actually, according to the random walk theory on networks [22], the probability of the arrival of a walker at a node of degree $d$, $pa$, is given by $pa = \frac{d}{2m}$, where $m$ is the total number of edges in the network. Therefore, when $C = 1$, the value of $rp \times pa$ is the same regardless of the value of $d$. That is, the best storage load balancing would be obtained when $C = 1$. However, in order to reduce the number of hops, the value of $C$ has to be set to be greater than one, and the appropriate value of $C$ would depend on $m$, which is related to network size and topology. Thus, $C$ is demonstrated to be a parameter for exploring the trade-off points between storage load balancing and search performance.

### 4.4 Experimental Results

We experimentally examine the performance of QR using the simulation model of a P2P network and the evaluation criteria explained in the previous section.

We use five replication probabilities for QR and PRR: 100%, 80%, 60%, 40%, and 20%. If the replication probability is the same for the two methods, then the number of times that replicas of the requested files are created would be approximately the same for the two methods. In addition, the replication probability of 100% should yield the smallest number of hops for both methods.

On the other hand, as mentioned in the previous section, the appropriate value of $C$ of RPID that yields good load balancing as well as fewer hops should depend on the network configuration. Therefore, we will use various values of $C$, ranging from 1 to 625, which are the minimum and the maximum degrees of the peers in the network used. Actually, RPID with $C = 625$ is the same as Path Replication.

For one replication probability or one value of $C$, we independently run the simulation model with fixed topology 20 times and show the result as the average of 20 runs.

The observed data is shown in Table 2, in which only the results of RPID for $C = 1, 10$, and 20 are shown. According to the preliminary experimental results, the average number of hops reaches a minimum at approximately $C = 20$, starting from $C = 1$. Meanwhile, the values of $C$ of approximately 20 provide the best load balancing among the values of $C$ that yield almost the minimum average number of hops. Therefore, the values of $C$ of approximately 20 can be regarded as the best among the values of $C$ that yield almost the minimum average number of hops.

Examples of lines that represent the storage loads of the three replication methods used herein are shown in Fig. 6, where the replication probability for QR and PRR is 100% and the value of $C$ in RPID is 20.

### 4.5 Discussion

We can observe from Table 2 that all of the replication methods used herein have the performance trade-off. That is to say, if the load balancing performance becomes higher, the search performance becomes lower and vice-versa. So, our focus here is the degree of the performance trade-off induced by each replication method. In a situation such that a certain replication method yields better load balancing performance (or better search performance) than other methods when the search performance (or the load balancing performance) of all of the methods are similar, we say that the replication method yields a *better trade-off point* than the others. We will use the phrase 'a better trade-off point' as this meaning hereinafter.

We first compare the result of QR with that of PRR. According to Table 2, we can see that QR is superior to PRR in terms of storage load balancing for any replication probability used, and that the average number of hops for initially and additionally distributed files are approximately the same. Therefore, we can say that QR yields better trade-off points than PRR.

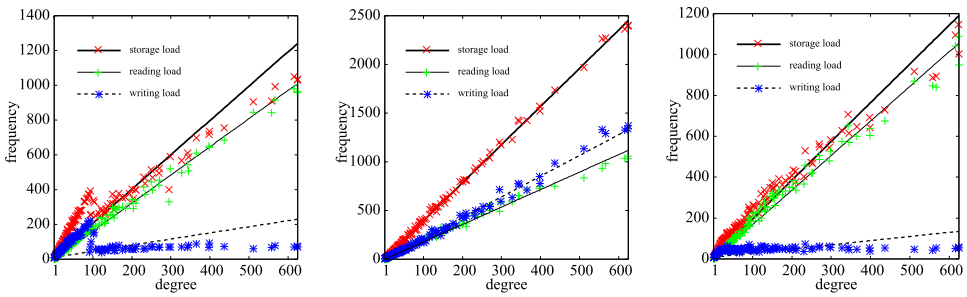
When these two methods have the same value of RP, the values of NW for them are almost the same, that is, the numbers of times of replications are almost the same between them. We can, therefore, say that comparison of the results between them using the same value of RP is fair. This simply suggests that the method of placing replicas of files over the networks is significant for storage load balancing among peers when the total number of replicas of files is fixed.

The difference in storage load between the two methods is caused by the difference in writing load between the two methods. According to Figs. 6 (a) and (b), while PRR gave a linear increase in the number of both writing and reading files with the degree of peers, QR caused the number of file writes in peers of degree greater than approximately 100 to be approximately equal, although the increase in the number of file writes in peers of degree less than approximately 100 was linear.

This discontinuous increase in the number of file writes

**Table 2** Experimental results. Each result represents the averaged result over 20 independent runs of the simulation model. RP is the replication probability. $C$ is the parameter of RPID. SL is the storage load. RL is the reading load. WL is the writing load. HI is the average number of hops for initially distributed files. HA is the average number of hops for additionally distributed files. NW is the total number of times of writing files. NF is the total number of files in the network.

| | RP | SL | RL | WL | HI | HA | NW | NF |
|---|---|---|---|---|---|---|---|---|
| QR | 100% | 2.008 | 1.654 | 0.354 | 2.186 | 2.145 | 116665 | 91812 |
| | 80% | 1.961 | 1.671 | 0.289 | 2.201 | 2.168 | 104957 | 87019 |
| | 60% | 1.922 | 1.695 | 0.226 | 2.232 | 2.191 | 92994 | 81494 |
| | 40% | 1.872 | 1.711 | 0.161 | 2.291 | 2.291 | 80965 | 75222 |
| | 20% | 1.789 | 1.692 | 0.096 | 2.510 | 2.556 | 68578 | 67526 |
| PRR | 100% | 3.946 | 1.812 | 2.133 | 2.251 | 2.221 | 117613 | 76336 |
| | 60% | 3.218 | 1.921 | 1.297 | 2.294 | 2.269 | 92048 | 70059 |
| | 20% | 2.658 | 2.190 | 0.467 | 2.452 | 2.424 | 65923 | 60830 |
| | $C$ | SL | RL | WL | HI | HA | NW | NF |
| RPID | 20 | 1.929 | 1.726 | 0.202 | 2.221 | 2.170 | 89596 | 79038 |
| | 10 | 1.754 | 1.637 | 0.117 | 2.361 | 2.369 | 87306 | 82136 |
| | 1 | 1.124 | 1.115 | 0.009 | 3.664 | 3.764 | 81478 | 82552 |




(a) QR (RP: 100%). (b) PRR (RP: 100%). (c) RPID ($C$ = 20).

**Fig. 6** Examples of lines representing storage load.

occurs as a result of the switching cases using Eq. (1) in Sect. 3.2. According to a preliminary experiment, the average degree of peers adjacent to the peer of interest is approximately 100. In addition, according to the random-walk theory mentioned before, a peer was on the previous successful search paths should be proportional to the degree of the peer. Therefore, peers meeting Case 2 described in Sect. 3.2 should have degree greater than approximately 100, and actually the discontinuous occurs around degree of 100.

The ability to make the number of file writes in peers with degrees of greater than approximately 100 equal is the most important concept in storage load balancing. This writing load balancing occurs as a result of Eq. (2) in Sect. 3.2. However, if we change the condition to determine the degree that divides all of the peers into two replication strategy groups, we could obtain results different from those obtained here.

Next, Table 2 and Fig. 6 (c) show that RPID achieved good writing load balancing. If we compare the results of QR, PRR, and RPID when the values of NW for them are similar to each other, for example, when the values of RP for QR and PRR are 60% and the value of $C$ for RPID is 20, we can see that their search performance are almost the same, and the performance of access load balancing of QR and RPID are similar but that of PRR is inferior to that of QR and RPID. Therefore, we can say that QR and RPID yield better trade-off points than PRR and also that QR and RPID are similar in terms of ability in yielding better trade-off points. In addition, QR with the replication probability of 80% or 100% yields slightly better search performance than RPID with $C = 20$, but slightly worse performance of access load balancing than RPID with $C = 20$.

Since RPID makes a replica of a requested file in peers on the present successful search path with a probability that is inversely proportional to the degree of peers, the number of file writes of RPID is expected to be approximately the same in every peer in the case of $C = 1$, when the probability of query arrival at a peer is proportional to the degree of the peer. However, RPID should require $C$ to be set appropriately, for example as $C = 20$, in order to handle the increase in the average number of hops by making peers of middle and high degrees have a sufficient number of files.

To verify this, we conducted an experiment with another power-law network topology with the same number of nodes as that in the simulation model described earlier, but with fewer edges (approximately 13,000) than that in the simulation model (20,000). The experimental results are shown in Table 3. Since we have confirmed from preliminary simulation experiments that the average number of hops does not vary much in the case that $C$ is greater than or equal to 5, we show only the results of the values of $C$ up to 10 in Table 3.

Table 3 shows that the appropriate value of $C$ for this network topology was approximately 5, which yields the

**Table 3** Experimental results used to verify that an appropriate value of $C$ in RPID depends on network topology. RP is the replication probability. $C$ is the parameter of RPID. SL is the storage load. RL is the reading load. WL is the writing load. HI is the average number of hops for initially distributed files. HA is the average number of hops for additionally distributed files.

| | RP | SL | RL | WL | HI | HA |
|---|---|---|---|---|---|---|
| QR | 100% | 4.12 | 3.27 | 0.84 | 3.42 | 3.43 |
| | 80% | 4.01 | 3.30 | 0.70 | 3.48 | 3.56 |
| | 60% | 3.89 | 3.34 | 0.54 | 3.57 | 3.68 |
| | 40% | 3.78 | 3.38 | 0.39 | 3.78 | 3.94 |
| | 20% | 3.60 | 3.36 | 0.24 | 4.25 | 4.48 |
| | $C$ | SL | RL | WL | HI | HA |
| RPID | 10 | 4.70 | 3.59 | 1.11 | 3.53 | 3.38 |
| | 5 | 3.96 | 3.34 | 0.61 | 3.56 | 3.48 |
| | 2 | 3.04 | 2.81 | 0.22 | 4.09 | 4.17 |
| | 1 | 2.38 | 2.31 | 0.07 | 4.69 | 4.81 |

best load balancing (the smallest value of $SL$) among values of $C$ that yield almost the minimum average number of hops, which is greater than or equal to 5. This result is different from that for the network topology described earlier (approximately $C = 20$). Since we cannot obtain the information on the whole unstructured P2P network, if we use RPID and attempt to achieve the minimum average number of hops while keeping better storage load balancing as much as possible, an adaptive mechanism for changing the value of $C$ should be embedded in the network so as to achieve that.

Meanwhile, QR also needs to set an appropriate replication probability for a given network. However, it is not as difficult as setting the value of $C$ for RPID. As shown in Tables 2 and 3, the smaller the replication probability of QR is, the lower the search performance is and the higher the storage load balancing is. That is to say, even if the network configurations as a network topology are changed, QR with the replication probability of 100% always yields the best load balancing performance among all of the possible replication probabilities that yield the minimum average number of hops.

Actually, as shown in Tables 2 and 3, QR and RPID are similar in terms of ability in yielding better trade-off points. For instance, we can see from Table 3 that QR with the replication probability of 80% and RPID with $C = 5$ have similar search performance and performance of access load balancing. However, the advantage of QR over RPID is that as mentioned above, whatever the network configurations are, we can easily understand the relationship between the parameter value of QR and the performance of QR and actually set the parameter value based on this understanding. To obtain the minimum average number of hops while achieving better storage load balancing as much as possible, at least we can more easily set the parameter value of QR than that of PRID.

Furthermore, RPID is basically effective only when a random-walk based query forwarding method is used in the first place. If we use other types of query forwarding methods, it would be difficult to obtain probability with which each peer is on successful search paths. Meanwhile, QR is expected to achieve at least local storage load balancing even for other types of query forwarding method, because its strategy is to create replicas of files in peers of low load adjacent to peers of high load based on query trails generated by the query forwarding methods.

## 5. Method for Balancing the Load of Reading Files from Storage

In the previous section, we proposed the first type of cooperative behaviors of peers, which is a replication method for exploring better trade-off points between storage load balancing and search performances. When using a random-walk-based query forwarding method, the probability of query arrival to a peer is proportional to degree of the peer. Therefore, even if the load of writing files in storage is balanced among peers, and consequently every peer holds the same number of files, the number of times that a file is read from the storage of a peer results in the probability being proportional to degree of the peer. If load balancing of reading files from storage among peers can be achieved, we can expect further storage load balancing among peers. In this section, we propose a method for balancing the load of reading files from storage among peers in return for a slight degradation in search performance, which is the second type of cooperative behaviors of peers.

### 5.1 Local Control of Providing Files by Peers

The proposed method for balancing the load of reading files from storages requires a peer to determine whether it provides a requested file to a peer making a query based on comparison of the load between the peer and its neighbors. Here, we also estimate the state of loads of peers based on query trails as the replication method presented in Sect. 3. The query trails used here are the points, which is one of the two types of query trails described in Sect. 3.

The procedures of the proposed method are as follows. First, the method obtains a value of $R_c \in [-1, +1]$ by Eq. (3), which contains the total number of points that a peer of interest assigned to its neighboring peers, $r_f$, and that its neighboring peers assigned to the peer of interest, $r_a$.

$$R_c = \begin{cases} 1 - \frac{r_a}{r_f} & \text{if } r_f \geq r_a, \\ \frac{r_f}{r_a} - 1 & \text{otherwise.} \end{cases} \quad (3)$$

The load that the peer of interest assigned to its neighboring peers is represented by $r_f$ and the load that its neighboring peers assigned to the peer of interest is represented by $r_a$. Therefore, the smaller the value of $R_c$, the higher the load of the peer of interest.

Second, using $R_c$ below, the method determines whether a peer holding a requested file provides a file according to the rules, in which $T_r$ is a parameter of the method. This rule is illustrated in Fig. 7.

- If $R_c \geq T_r$, then the peer provides the requested file.

- If $R_c < T_r$, then the peer refuses to provide the requested file.

The proposed method for balancing the load of reading files from storages among peers is composed of Eq. (3) and the above rules. We call the proposed method the Query-trail-based File Provision, or QFP.

### 5.2 Simulation Results

We combine the proposed QFP with QR proposed in Sect. 3 and compare this combination method with QR through simulations. As values of $T_r$, which is the parameter of QFP, we used nine different values, −0.1, −0.2, −0.3, −0.4, −0.5, −0.6, −0.7, −0.8, and −0.9. The replication probability, which is the parameter of QR, is set to 100%. The simulation model and its settings are the same as those shown in Sect. 3. For each value of $T_r$, the simulation model was run 20 times, and the averaged results over 20 independent runs were obtained with respect to storage load balancing and search performance. These results are actually six types of points, (SL, HI), (WL, HI), (RL, HI), (SL, HA), (WL, HA), and (RL, HA), where SL, WL, RL, HI, and HA were shown in Table 2. These six types of points are plotted in Fig. 8. Figure 8 (a) shows the average number of hops for initially distributed files, which are (SL, HI), (WL, HI), and (RL, HI). Figure 8 (b) shows the average number of hops for additionally distributed files, which are (SL, HA), (WL, HA), and (RL, HA). In addition, Fig. 8 (c) shows the example graph representing storage load for the case of $T_r = -0.4$.

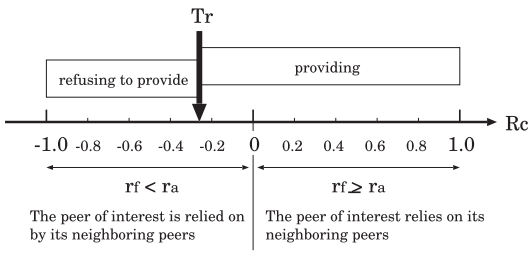



**Fig. 7** The rule to determine whether a peer provides a requested file.

According to Fig. 8, with the increase of $T_r$, the combination method of QR and QFP greatly improved the storage load balancing performance by balancing well the load of reading files from storage among the peers. Although increasing the value of $T_r$ caused an increase in the total number of file written to storage, the performance of balancing the load of writing files to storage was not reduced significantly because the performance of QR of load balancing of writing files to storage is quite high.

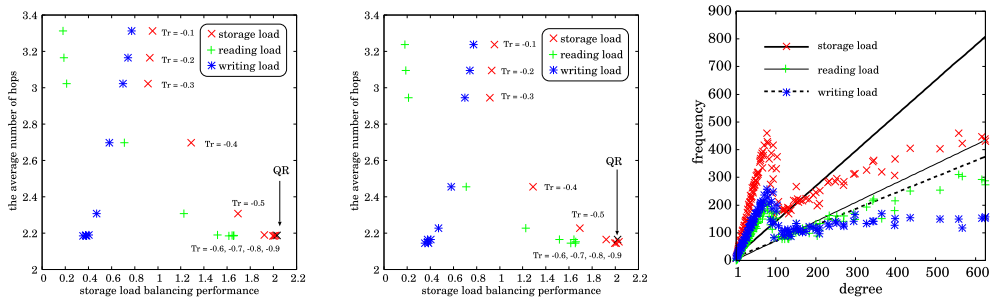
In actual P2P file sharing networks, the file search time that users can accept and the amount of traffic that networks can accept are different. Therefore, achieving higher storage load balancing with degradation of search performance is not always good for users. However, if the acceptable file search time and the acceptable amount of traffic are large, the use of QFP can improve storage load balancing, and the degree of improvement of the storage load balancing performance can be regulated by the value of $T_r$. The value of $T_r$ should be determined by network operators of actual P2P file sharing networks, considering the situations of the networks. Therefore, it is impossible to determine an appropriate value of $T_r$ for any situation of the networks. However, in this case, a replication method is required that yields better load balancing of writing files in storages together with QFP.

## 6. Evaluation in Dynamic and Heterogeneous P2P Environments

In the evaluation presented in Sects. 4 and 5, we assumed a static and homogeneous P2P network in the simulation model, though degree is different among the peers. In this section, in contrast to Sects. 4 and 5, we assume a dynamic and heterogeneous P2P network in the simulation model and then evaluate QR and the combination method of QR and QFP compared to PRR through simulations.

### 6.1 P2P Simulation Model

The number of peers in the simulation model used herein




(a) Relationship between storage load and average number of hops for initially allocated files when using a variety of values of $T_r$.

(b) Relationship between storage load and average number of hops for additionally allocated files when using a variety of values of $T_r$.

(c) Examples of lines representing storage load when using $T_r = -0.4$.

**Fig. 8** Simulation results obtained by examining the combination method of QR and QFP for peers with the same storage capacity.

is set to be 10,000 as in Sects. 4 and 5. Every peer which is present in the network at a certain moment conducts a file search only once. A unit time is defined as time which is required for all of peers present in the network at a certain moment to finish their file searches. RP (the replication probability) for PRR, QR, and the combination method of QR and QFP is set to be 100%. The other configurations of the simulation are mentioned below.

- Topology
  According to [30], the Gnutella which is one of the representative unstructured P2P networks include peers which frequently leave and join the network as well as ones which stably stay in the network. Therefore, if time for which the topology information is gathered is long, peers which did not exist in the network at the same time are incorrectly included in one network topology due to frequent participation and departure of peers. In addition, it is shown that the shapes of distribution of degree of peers depend on time for which the topology information is gathered. Especially, when the time for gathering is long, a power law network as reported in [29] appears, and otherwise, the power law network does not appear.
  In this section, we consider a dynamic topology generation method which is involved in the P2P simulation model based on the fact presented in [30] that there are peers which frequently leave and join the network and ones which stay in the network stably. We will examine in the future work if the dynamically generated topologies in this section fit the results reported in [30].
  In the topology generation considered herein, each peer, at every unit time, joins the network according to probability which was given to it in advance. The probability to determine if the $i$-th peer joins the network at every time, $pp_i$, is a uniform random real number whitin $[0.1, 1.0]$. This realizes the existence of peers that frequently leave and join the network as well as stay in the network for a long period of time. We have confirmed by preliminary simulations that about 5,500 out of 10,000 peers join the network at any moment. In addition, a peer that has decided to join the network makes an undirected link to other peer at any time. As a result, the total number of links at a certain moment is equal to the number of peers that join the network at that moment.
  When a peer joins the network for the first time, the peer randomly selects a peer to which the joining peer will link from among all of the peers present in the network at that moment. Next, when the joining peer leaves the network and then rejoins the network, it tries to link to the same peer that it linked to before leaving the network. If the same peer is not in the network, the node randomly selects a peer to which the joining node will link from among all of the peers present in the network at that moment.
- Query Distribution
  In the simulations herein, 1,000 types of files are present in the network. Each type of file is held by 10 peers. Serial numbers are assigned to all of the 10,000 peers, and the $10(k-1)$-th to the $10k$-th peers have the $k$-th type of file initially, where $k = 1, 2, \cdots, 1{,}000$. We consider here that one file consumes one unit of storage for record.
  It is said that query distribution which represents how frequently each file is queried to follows Zipf's law [31] in general. Zipf's law is represented by Eq. (4).
  $$f = kx^{-\alpha}, \tag{4}$$
  where $f$ is the query generation ratio, $x$ is the rank of popularity of a file, and $\alpha$ stands for the degree of imbalance of popularity among files. According to Zipf's law, a few files with high popularity have most accesses. In this section, we will decide the query generation ratio to files by following Zipf's law and set the parameter $\alpha = 1.0$ in Eq. (4).
  We simulate time-varying query distributions. The way to simulate this is that the 301st to the 1000th types of files are queried according to Zipf's law from time 1 to 25 and the first to the 700th types of files are queried according to Zipf's law from time 26 to 50. The time-varying query distributions used herein is shown in Fig. 9.
- Storage Capacity
  Storage capacity of each peer is proportional to its own probability to join the network. The storage capacity for the $i$-th peer, $C_i$, is represented by Eq. (5).
  $$C_i = 100 \times pp_i, \tag{5}$$
  where $C_i \in [10, 100]$.

## 6.2 Evaporation of Query Trails

We need to be able to estimate the storage load of peers at any moment in a dynamic P2P environment considered herein as precisely as possible. For this purpose, we consider that the peers evaporate their query trails periodically.

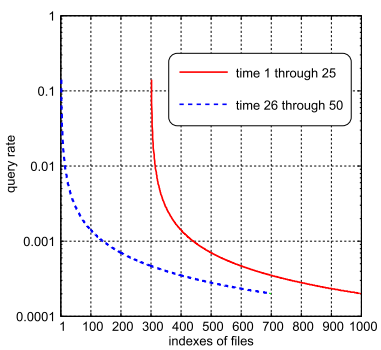
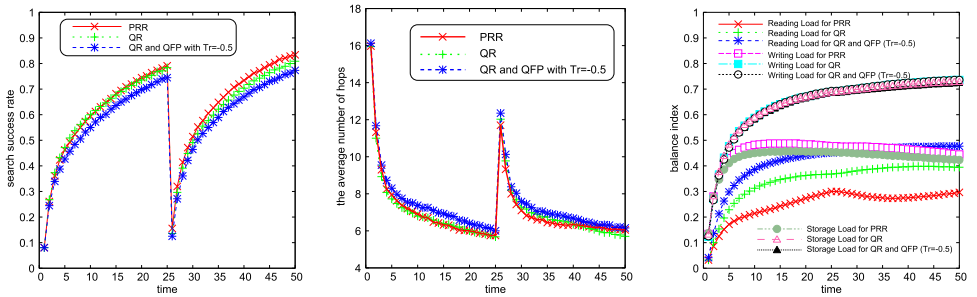



**Fig. 9** Dynamic change in query distribution. The solid line represents query distribution for time 1 through 25. The dashed line represents query distribution for time 26 through 50.




(a) Success rate of search. (b) The average number of hops. (c) The balance indexes for reading, writing, and storage loads.

**Fig. 10** Simulation results of PRR, QR, and the combination method of QR and QFP with $T_r = -0.5$ in the dynamic and heterogeneous P2P environment. The results are the average over 20 independent runs.

Concretely, the peers make the current values representing the strength of query trails a half every 10 unit times. That is, points that the peers assigned to their neighbors and the number of times that the peers were on the previous successful search paths are reduced to a half every 10 unit times.

When peers leave the network, the links of the peers disappear and points that the peers assigned to their neighbors are reset to be zero. The number of times that the peers were on the previous successful search paths are not varied at the timing of departure of the peers.

### 6.3 Evaluation Criteria

We use the Balance Index [13] as an index of storage load balancing performance. The Balance Index, $\delta$, is defined by Eq. (6). The closer the value of $\delta$ is to 1, the more highly access load is balanced among peers. On the other hand, the closer the value is to $1/N$, the more highly access load is biased to particular peers.

$$\delta = \frac{(\sum_{i=1}^{N} X_i)^2}{(N \sum_{i=1}^{N} X_i^2)}, \tag{6}$$

where $X_i$ is the number of accesses to peer $i$ and $N$ is the total number of peers. The number of accesses refers to the number of times of reading files from the storage and that of writing files in the storage. As $X_i$, we consider three kinds of values, which are the number of times of reading files from the storage (RL: Reading Load), that of writing files in the storage (WL: Writing Load), and the sum of those of reading files from the storage and of writing files in the storage (SL: Storage Load). The Balance Index is calculated at every unit time using RL, WL, and SL of all of the peers including peers which are not present in the network at a certain time.

We use the success ratio of file search as an index of search performance, as well as the average number of hops for successful searches. Those values are also calculated at every unit time.

### 6.4 Results

Figure 10 shows the success ratio of file search, the average number of hops for successful searches, and the Balance Index for PRR, QR, and the combination of QR and QFP with $T_r = -0.5$. The results are the average over 20 independent simulation runs.

We can see from Fig. 10 (a)(b) that all of the methods yielded almost the equal search performance. Furthermore, we can also see that QR, and the combination of QR and QFP with $T_r = -0.5$ yielded better storage load balancing performance than PRR. These results suggest that QR, and the combination of QR and QFP achieve both of storage load balancing and high search performance even in dynamic and heterogeneous P2P environments.

## 7. Conclusions

We presented two types of query-trail-mediated cooperative behavior of peers for exploring the performance trade-off in unstructured P2P networks. The origin of storage load imbalance among peers in the present paper is random-walk-based query forwarding on a given network topology. We evaluated these two types of query-trail-mediated cooperative behavior of peers assuming a static and almost homogeneous P2P environment through simulations. Simulation results showed that the first type of cooperative behavior provides equal or improved ability to explore trade-off points between storage load balancing and search performance, without the need for fine tuning parameter values, compared to replication methods that require fine tuning of their parameters values. In addition, the combination of the second type and the first type of cooperative behavior yields better storage load balancing performance with little degradation of search performance. Furthermore, we evaluated the two types of query-trail-mediated cooperative behaviors of peers assuming a dynamic and heterogeneous P2P environment through simulations. We showed that the two types of cooperative behaviors of peers provide a high capability to explore better trade-off points between storage load balancing and search performance even in the dynamic and

heterogeneous P2P environment.

In the present paper, the main cause of storage load imbalance among peers was the use of the random-walk-based query forwarding method. However, even for other types of query forwarding methods, at least local storage load balancing was expected by applying a strategy that creates replicas of files in peers of low load adjacent to peers of high load based on query trails generated by the query forwarding methods. Furthermore, if we first reveal which peers adjacent to peers of high load queries are frequently forwarded from the peers with high load in consideration of the characteristics of used query forwarding methods and then design rules on where to place replicas of files, we could reduce the degradation of the search performance. In the case of using random-walk-based query forwarding methods, a query that arrived at a peer of high load is, with equal probability, forwarded to its neighboring peers. In addition, if we introduce local control of providing files by peers together with file replication methods such as those described above, we can expect higher storage load balancing performance with little degradation in search performance.

## Acknowledgments

The present study was supported by the Japan Society for the Promotion of Science through a Grant-in-Aid for Young Scientists (B) (22700077).

### 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 & Tutorials, vol.7, no.2, pp.72–93, 2005.
[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: Scalable peer-to-peer lookup service for internet applications," Proc. ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp.149–160, San Diego, CA, USA, Aug. 2001.
[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," Proc. ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp.161–172, San Diego, CA, USA, Aug. 2001.
[4] A. Rowstron and P. Druschel, "Pastry: Scalable decentralized object location and routing for large-scale peer-to-peer systems," Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp.329–350, Heidelberg, Germany, Nov. 2001.
[5] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for wide-area location and routing," Technical Report UCB CSD-01-1141, U.C.Berkeley, Berkeley, CA, USA, 2001.
[6] "LimeWire." http://www.limewire.com/en
[7] I. Clarke, O. Snadberg, B. Wiley, and T.W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," Proc. Workshop on Design Issue in Anonymity and Unobservability, 2000.
[8] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," Proc. ACM SIGCOMM 2002, 2002.
[9] Q. Lv, P. Cao, E. Cohen, S. Li, and K. Shenker, "Search and replication in unstructured peer-to-peer networks," Proc. 16th International Conference on Supercomputing, pp.84–95, 2002.
[10] L. Rong, "Multimedia resource replication strategy for a pervasive peer-to-peer environment," J. Comput., vol.3, no.4, pp.9–15, April 2008.
[11] S.M. Thampi and C.S. K, "Review of replication schemes for unstructured p2p networks," Proc. IEEE International Advance Computing Conference IEEE (IACC'09), pp.794–800, Patiala, India, March 2009.
[12] Y. Drougas and V. Kalogeraki, "A fair resource allocation algorithm for peer-to-peer overlays," 25th IEEE INFOCOM Conference, pp.3085–90, 2006.
[13] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," Computer Networks and ISDN Systems, vol.17, no.1, pp.1–14, 1989.
[14] H. Yamamoto, D. Maruta, and Y. Oie, "Replication method for load balancing on distributed storages in P2P networks," IEICE Trans. Inf. & Syst. - Special Section on New Technologies and their Applications of the Internet III, vol.E89-D, no.1, pp.171–181, Jan. 2006.
[15] D. Choffnes and F. Bustamante, "Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems," Comput. Commun. Review, vol.38, no.4, pp.363–74, Oct. 2008.
[16] Y.-M. Chiu and D.Y. Eun, "Minimizing file download time in stochastic peer-to-peer networks," IEEE/ACM Trans. Netw., vol.16, no.2, pp.253–66, April 2008.
[17] E. Pournaras, G. Exarchakos, and N. Antonopoulos, "Load-driven neighbourhood reconfiguration of Gnutella overlay," Comput. Commun., vol.31, no.13, pp.3030–9, 15 Aug. 2008.
[18] M. Srivatsa, B. Gedik, and L. Liu, "Large scaling unstructured peer-to-peer networks with heterogeneity-aware topology and routing," IEEE Trans. Parallel Distrib. Syst., vol.17, no.11, pp.1277–93, Nov. 2006.
[19] W.-C. Liao, F. Papadopoulos, and K. Psounis, "Performance analysis of BitTorrent-like systems with heterogeneous users," Performance Evaluation, vol.64, no.9-12, pp.876–91, Oct. 2007.
[20] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?," IEEE J. Sel. Areas Commun., vol.25, no.9, pp.1678–94, Dec. 2007.
[21] K. Ohnishi, H. Yamamoto, K. Ichikawa, M. Uchida, and Y. Oie, "Storage load balancing via local interactions among peers in unstructured P2P networks," International Workshop on Peer-to-Peer Infromation Management (P2PIM2006), CD-ROM, 8 pages, 2006.
[22] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
[23] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," IEEE Trans. Syst. Man, Cybern. B, Cybern. vol.26, no.2, pp.29–41, 1996.
[24] M. Dorigo and L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol.1, no.1, pp.53–66, 1997.
[25] G.D. Caro and M. Dorigo, "AntNet: Mobile agents for adaptive routing," Proc. 31st Hawaii Int'l Conf. on System Sciences, pp.74–83, 1998.
[26] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," Proc. IEEE Infocom 2002, pp.638–647, 2003.
[27] A.L. Barabasi and R. Albert, "Emergence of scaling in random networks," Science, vol.286, pp.509–512, 1999.
[28] R. Albert and A.L. Barabasi, "Topology of evolving networks: Local events and universality," Phys. Rev. Lett., pp.5234–5237, 2000.
[29] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the gnutella network," IEEE Internet Comput., vol.6, no.1, pp.50–57, 2002.
[30] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing unstructured overlay topologies in modern P2P file-sharing systems," IEEE/ACM Trans. Netw. (TON), vol.16, no.2, pp.267–280, 2008.
[31] L. Adamic and B. Huberman, "The nature of markets in the world wide web," Quarterly Journal of Electronic Commerce, vol.1, no.1, pp.5–12, 2000.

**Kei Ohnishi** received the B.E., M.E. and D.E. degrees from Kyushu Institute of Design, Japan in 1998, 2000, and 2003, respectively. He worked as a postdoctoral researcher for University of Illinois at Urbana-Champaign, Kyushu Institute of Technology, and Human Media Creation Center/Kyushu. Since October 2007, he has been an associate professor at Kyushu Institute of Technology. His research interests include P2P networks and soft computing techniques. He is a member of IEEE, IPSJ, and SOFT.

**Hiroshi Yamamoto** received M.E. and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 2003 and 2006, respectively. From April 2006 to March 2010, he worked at FUJITSU LABORATORIES LTD., Kawasaki, Japan. Since April 2010, he has been an Assistant Professor in the Department of Electrical Engineering, Nagaoka University of Technology. His research interests include computer networks, distributed applications, and networked services. He is a member of the IEEE.

**Masato Uchida** received B.E., M.E. and D.E. degrees from Hokkaido University, Sapporo, Hokkaido, Japan in 1999, 2001 and 2005, respectively. In 2001, he joined NTT Service Integration Laboratories, Tokyo, Japan. Since August 2005, he has been an Associate Professor in Network Design Research Center, Kyushu Institute of Technology.

**Yuji Oie** received B.E., M.E. and D.E. degrees from Kyoto University, Kyoto, Japan in 1978, 1980 and 1987, respectively. From 1980 to 1983, he worked at Nippon Denso Company Ltd., Kariya. From 1983 to 1990, he was with the Department of Electrical Engineering, Sasebo College of Technology, Sasebo. From 1990 to 1995, he was an Associate Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka. From 1995 to 1997, he was a Professor in the Information Technology Center, Nara Institute of Science and Technology. Since April 1997, he has been a Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer communication networks, high speed networks, and queueing systems. He is a fellow of the IPSJ and a member of the IEEE.