An Effective Lookup Strategy for Recursive and Iterative Lookup on Hierarchical DHT

Tomonori Funahashi [†], Yoshitaka Nakamura [‡], Yoh Shiraishi [‡], and Osamu Takahashi [‡]

[†] Graduate School of Systems Information Science, Future University Hakodate, Japan
[‡] School of Systems Information Science, Future University Hakodate, Japan
{g2111032, y-nakamr, siraisi, osamu}@fun.ac.jp

Abstract - Recursive and iterative lookups on the performance of distributed hash table (DHT) are deteriorated by churn when nodes leave the network. When churn occurs infrequently, recursive lookup outperforms iterative lookup, but back when churn occurs frequently, the opposite is the case. Therefore, optimal lookup needs recursive and iterative lookups to be separated by the frequency of churn. We propose a lookup strategy that separates recursive and iterative lookups by the churn rate. However, a common DHT makes it difficult establish the neighboring churn rate. Hierarchical DHT takes into consideration the reliability of nodes to ascertain the churn rate, but it uses only a lookup strategy, recursive or iterative in the DHT. We believe that each lookup strategy should be used that match the churn rate in hierarchical DHT. Therefore, we compared our lookup strategy with both recursive and iterative lookup on hierarchical DHT.

Keywords: Recursive lookup, Iterative lookup, Hierarchical DHT

1 INTRODUCTION

Peer-to-Peer (P2P) is communication in which each node is equal and various values are dispersed throughout the network. Therefore, distributed hash table (DHT) is an efficient lookup technology in P2P. DHT can discover values with low numbers of hops in large networks. Examples of DHT-based P2P include Chord[1], Kademlia[2], and Pastry[3]. Even if DHT uses the same algorithm as Chord or has routes on the same lookup path, their communication methods are defined differently. Its methods are known to be recursive and iterative lookups[4]. These lookups have different lookup latencies and numbers of messages. Recursive lookup, which has low latency, is generally satisfactory. However, the performance of these lookups deteriorates due to churn where nodes leave the network. In addition, recursive lookup performs worse than iterative lookup. Therefore, optimal lookup needs recursive and iterative lookups to be separated by the system churn rate. However, flat normal DHT it is not structured to consider the feature of nodes, e.g. the churn of nodes. For this reason, it is difficult to establish the system churn rate.

There is a structure called hierarchical DHT[8][9] that enables DHT to be used efficiently. This structure can separate a number of clusters depending on needs. A hierarchical DHT has been developed with advanced features that take into consideration how reliable nodes are[10]. This has a clustering method that establishes the reliability of nodes. Thus, the reliability of each cluster is approximately established.

We propose applying an optimal lookup strategy to each cluster on hierarchical DHT that considers the reliability of nodes and separates recursive and iterative lookups efficiently.

2 RELATED WORK

2.1 Chord

Chord is a DHT algorithm that takes into consideration the hash space as a space like a ring and sets nodes an identifier called the node ID with the hash function. Keys are calculated similarly with this function. Of the nodes arriving in a network, the node just behind a node is called a successor node, and the one just before a node is called a predecessor node. Nodes keep the neighbor as a successor list that has a number of successor nodes and a finger table that can route efficiently to the routing table. Chord completes lookup with path length $O(\log N)$ when N is the number of all nodes. The state of these nodes is the previous state obtained by churn and failure. For this reason, Chord is implemented as a stabilization process to accurately retain the state of neighbor nodes. This is a process where nodes ask nodes in the routing table. In addition, it is executed at regular intervals.

2.2 Lookup strategy

Recursive lookup is a lookup method that uses an originator node that demands value requests lookup other nodes. However, iterative lookup is where the originator controls lookup to ask other nodes about candidates for the next hop. Figure 1 outlines the shape of each lookup on Chord when there are three hops (path length).

The originator in recursive lookup forwards a request message to a node that is closer to the destination (Figure 1 (1)). If a node in the next phase receiving a request message does not have the value that is purposed, it forwards the request message to a node that is closer to the destination than itself. This process is executed until the request message reaches the destination node (Fig. 1 (2), (3)). In contrast, the originator in iterative lookup receives a reply message to a request message after the message has been forwarded (Fig. 1 (1-2), (2-2)).

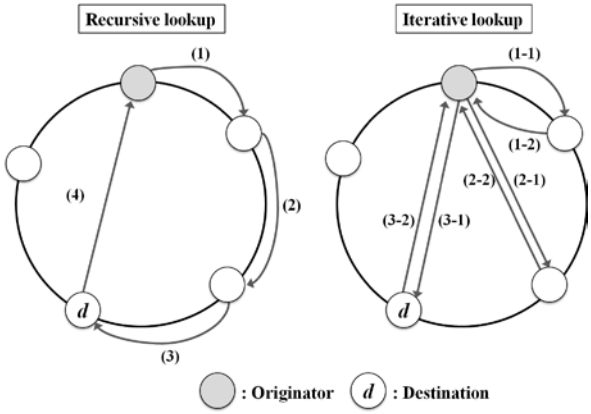

Figure 1: Recursive and iterative lookup strategy on Chord when path length = 3.

When a node that receives a request message does not have the value that is purposed, the reply message includes the addresses of nodes that are closer to the destination than itself. The originator forwards a request message to the destination node by using the address included in the reply message.

The performance of recursive and iterative lookups is affected in accordance with this communication method and churn where nodes leave the network. The system churn rate, which is the probability of which nodes on the network will leave the network determines the life-time of nodes. *R* is the defined life-time of a node and refers to the reliability of nodes. *R* varies between nodes. The cumulative distribution function[5] of exponential or Pareto distribution[6] is used as a function to define *R*. *R* makes known how often churn occurs in the system. *S* is defined as the time until nodes detect failure and repair the routing table of the node when churn or failure occurs. For this reason, *S* just means the interval in which the stabilization process is executed. Altogether, large *S* means that the stabilization process is seldom executed, but small *S* means that stabilization is executed often.

We also assumed that $E[R]$ and $E[S]$ were value expected for the *R* and *S* of neighbor nodes for a node. By using these parameters, *p* is defined as the probability of which next hop candidate node is alive in the network and the success of forwarding a request message, which is given by the following[7].

$$p = \frac{E[R]}{E[R] + E[S]} \tag{1}$$

When neighbor nodes are in a steady state when starting lookup and the originator is not executed to repair its own routing table, $E[S]$ approximates a fixed value. As a result, *p* depends on $E[R]$. In addition, large $E[R]$ means that neighbor nodes are alive for a long time, and this also means that churn is not likely to occur. In contrast, small $E[R]$ means that churn often occurs in neighbor nodes that have shorter life-times. That is, the churn rate is low when *p* is high and high when *p* is low. More specifically, *p* becomes a value that means the churn rate in the network when $E[S]$ approximates a fixed value.

The performance of recursive and iterative lookups are defined[7] by using churn rate *p* and latency of communication. First, we assume that the lookup path length is *l* and *t* is the latency for one hop. We also assume that physical links between nodes are not considered, and *t* is fixed. In addition, *T* is the time, which is timeout when nodes fail to forward messages by churn or failure. Here, timeout *T* is configured differently at each lookup. The originator in recursive lookup has to wait for responses to complete as lookup is completed. However, other nodes only forward request messages to the next hop node and are not concerned with the forwarded message. Therefore, *T* in recursive lookup is set to no less than the time to complete the entire lookup at only the originator. For this reason, T_r as the timeout in recursive lookup is configured as $T_r \geq (l+1)t$. The originator in iterative lookup similarly waits for a response from the next hop node point by point. Therefore, timeout is configured to no less than the time to wait for forward and reply. Consequently, T_i is the timeout in iterative lookup set by $T_i \geq 2\ t$. As a result, the expected latency of recursive lookup $E[RL]$ is defined in the following by these parameters.

$$E[RL] = (l+1)t + \frac{1-p^l}{p^l} T_r \tag{2}$$

The expected latency of iterative lookup $E[IL]$ is also defined in the following.

$$E[IL] = 2lt + \frac{1-p}{p} l T_i \tag{3}$$

In both recursive and iterative lookups, when *l* and *t* are fixed, *p* has a profound effect on performance. Figure 2 shows that an example of all expected latencies under different *p* when *l* and *t* are fixed values.

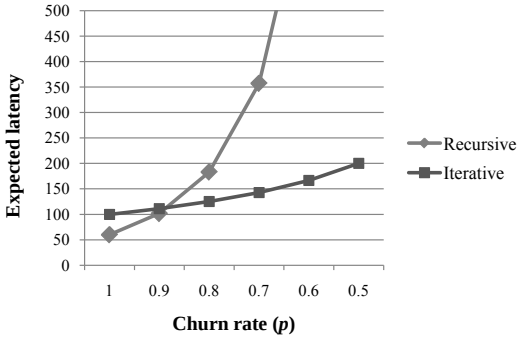

Figure 2: Expected latencies of recursive and iterative lookups under different *p*.

Moreover, T_r is much higher than T_i with this timeout setting. Thus, by using formula (2), the expected latency of recursive lookup increases especially when *p* is low. When *p* is low, on the other hand, iterative lookup does not have such high latency. However, when *p* is high, e.g. $p = 1$, this is higher than that of recursive lookup. Therefore, to improve the performance of recursive and iterative lookups, we need to determine the system churn rate.

2.3 Hierarchical DHT

Hierarchical DHT is a structure that divides a logical network configuration created by the DHT algorithm[8][9]. Figure 3 shows an example of a hierarchical DHT with two tiers in the Chord algorithm. Divided networks are called

top- and lower-level clusters. A top-level cluster is built by particular nodes called super nodes. Super nodes generally adopt strong nodes in the network, e.g., those with a great deal of high storage and high processing capacities that have been alive in the network for a long time, or those with wide bandwidth. Other normal nodes and a specific super node belong to the lower-level cluster. The super node provides normal nodes with routes to other clusters.

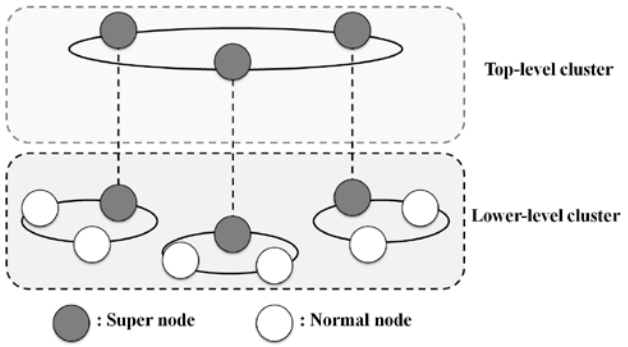

Figure 3: Example of two-tier hierarchical DHT.

Hierarchical DHT can speculate clusters where the destination of lookup belongs by comparing high m bits between the key and node ID. This m means the number of clusters in the hierarchical DHT by 2^m. When the high m bits of the key and a node ID are the same, the node forwards in the cluster. Otherwise, the node asks the super node of the cluster to forward, and the super node finds the destination cluster and super node address by using the key.

Hierarchical DHT has various features, i.e., to assemble normal nodes for their purpose and confine the effect of churn locally for neighbor nodes. An advanced study of hierarchical DHT found it to take into account the reliability of nodes[10]. This determines low-level clusters where normal nodes belong by using the interval from when they join to when they leave. The interval time is assumed by using a function, and this means that it is equivalent to R as the life-time of a node. The function in this study assembled nodes that had similar R in each cluster. In addition, a super node was selected as a node that had the highest R in the cluster. Nodes are clusters obtained by R in this way in hierarchical DHT that considers reliability. Therefore, $E[R]$ becomes high due to clustering nodes that have higher R, and this also decreases by using clustering nodes that have lower R. Here, we assume that the interval for the stabilization process is fixed at all nodes and nodes obtain $E[S]$, which is almost a fixed value. p is defined as $E[R]$ in formula (1), and so this differs specifically for each cluster. Therefore, the p of each cluster can be speculated, and we can consider the optimal performance of a system that is appropriate to p.

3 GOAL AND APPROACH

When p is low in recursive and iterative lookups, recursive lookup has an advantage, but when p is high, iterative lookup has an advantage. We select which lookup to uses by using the churn rate. This ensured that the expected latency of lookups was the best under any churn rate. Our goal was to demonstrate this. To speculate churn rate p, we noted hierarchical DHT took reliability into account. Hierarchical DHT determines clusters in which p is high or low as a result of clustering by the R of nodes. We focused on a structure where p was different for each cluster. And we believe that lookup strategies may be used to match the p of clusters. However, the hierarchical DHT uses one of the lookup strategies. Furthermore, we must consider that each message format in recursive and iterative lookups is different.

Here, we propose a strategy that changes over from one lookup to another by transforming the format of messages. We will explain how this strategy optimizes performance more than when only recursive or iterative lookup is used.

4 PROPOSED METHOD

4.1 System model

We propose that each cluster separates recursive and iterative lookups on hierarchical DHT to consider reliability. We used the Chord algorithm because it had various features, e.g., it had a simple structure and was scalable. We also noted the stabilization process for the formula (3). Although super nodes were adopted in the clusters, we assumed that super nodes would be adopted in the system. This meant that the R of super nodes had no relationship with the R in the clusters. Here, the R of super nodes is R_s, and that of other normal nodes is R_n. Clusters in assembled nodes that have low R_n, called lower clusters, use iterative lookup in the clusters because they have low p. However, clusters in assembled nodes that have high R_n, called higher clusters, use recursive lookup. For example, a top-level cluster built by a super node has R_s. R_s is relatively high approximately R in the system. Therefore, a top-level cluster uses recursive lookup. There are recursive and iterative lookups in the system for this reason. Here, it transforms from recursive into iterative and vice versa depending on the message format. This process is executed at super nodes. This provides the communication between higher and lower clusters.

All nodes have a routing table built by the Chord algorithm to structure hierarchical DHT. For example, that of the normal node includes normal nodes that belong to the same cluster and super nodes of the cluster. Also, super nodes have routing tables that included normal nodes belonging to the cluster and the super nodes of the top-level cluster.

4.2 Transformed process

There are request and reply messages in recursive and iterative lookups. Each message format is different due to the lookup strategy. For example, a reply message including next hop candidates is used in iterative lookup as a routing table. However, no reply messages are used in recursive lookup. Tables 1 and 2 indicate that both request and reply messages have to include information at least in recursive and iterative lookups.

Table 1: Information in request message.

	Identifier	Key ID	Address of originator	TTL
Recursive	○	○	○	○
Iterative		○		

Table 2: Information in reply message.

	Identifier	Next hop Candidates
Recursive	○	
Iterative		○

Recursive lookup can forward in parallel because it trusts other nodes with forwarding request messages. Messages have to include the address of the originator, the message identifier that determines what value is received for which request message, and the Time To Live (TTL), which is always set to forward request messages. This includes the message identifier. In iterative lookup, on the other hand, request messages do not have to include the address of the originator, identifier, or TTL because the originator controls the lookup. It only includes the key ID. However, reply messages must have some next hop candidates. Forwarding cannot continue because request and reply messages in both lookups are missing some necessary information.

By considering these differences, we implemented a transformed message format and lookup strategy. This transformed process particularly executes the transform from recursive to iterative and vice versa. It needs to be executed at all nodes on a flat DHT that does not have a hierarchy. However, the extent of the lookup strategy on hierarchical DHT is localized by clustering. For this reason, the transformed process is only executed at super nodes, which are contact points between clusters. The super nodes are confined to belong to lower clusters. They provide normal nodes with forwarding to top-level cluster and other clusters. Also, they provide other super nodes with forwarding to lower clusters. The flow for this operation of super nodes is outlined in Fig. 4.

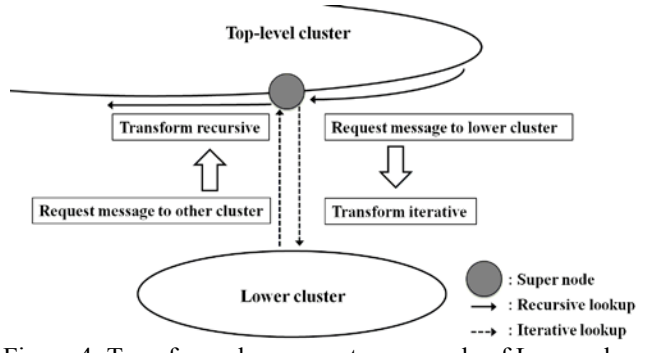

Figure 4: Transformed process at super node of Lower cluster.

When a super node receives a request message for iterative lookup from a normal node, if the destination is in another cluster, it creates a request message for recursive lookup from the subject matter of that message. However, the request message for iterative lookup does not include the identifier, the address of the originator, or TTL. For this reason, the super node creates a new identifier for the request message, and sets the TTL from the route. Also, the address of the originator is specified by the super node. Normal nodes do not read messages for recursive lookup because they do not transform from recursive into iterative message format. Therefore, super nodes provide the originator with a forwarding destination node and accept the reply message including the value with the transformed recursive into iterative message format.

However, when a super node belonging to a lower cluster receives a request message for recursive lookup, it can create a message for iterative lookup by only obtaining a key ID from the message. The value from the destination node similarly passes the super node, and the value of the transformed format is sent.

4.3 Lookup strategy

We propose that higher clusters use recursive lookup, and lower clusters use iterative lookup. Here, a top-level cluster is recognized as a higher cluster and uses recursive lookup. As a result, the pattern for lookup executed in the above transformed process is categorized as two patterns, (A) from the lower to the top-level cluster, and (B) from the higher to the lower cluster.

First, Fig. 5 shows an example of pattern (A).

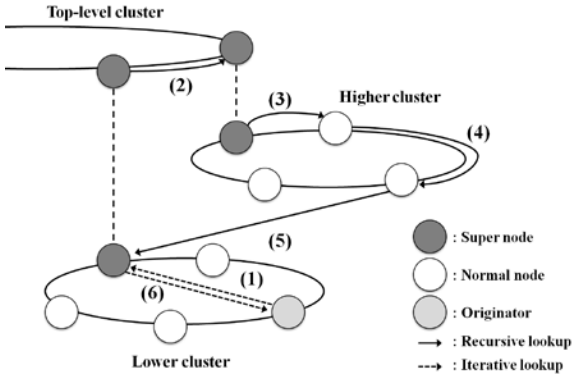

Figure 5: Lookup from lower to higher cluster.

The flow for lookup where request and reply messages are forwarded is indicated by the number in Fig. 5. In addition, request messages for iterative lookup are transformed into those for recursive lookup. First, the originator requests a super node to forward to another cluster with iterative lookup (Fig. 5 (1)). The super node transforms the message at the start, and starts recursive lookup. The lookup forwards to super and destination nodes (Fig. 5 (2)-(4)). Although the destination does not directly send the value to the originator, it sends the super node belonging to the originator (Fig. 5 (5)). The super node transforms the received message, and sends data to the originator (Fig. 5 (6)). We consider that this pattern shorten the latency of the entire lookup more than that with only iterative lookup because it uses recursive lookup at the part with low churn.

Second, Fig. 6 shows an example of pattern (B).

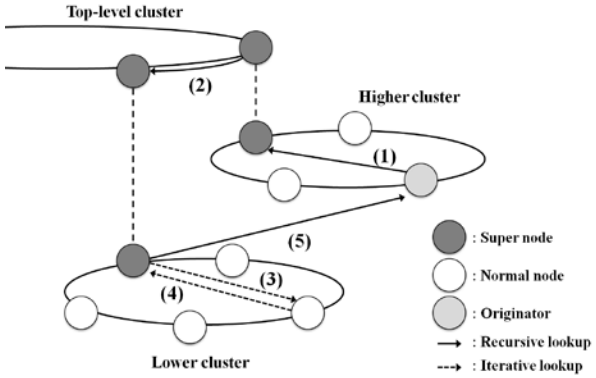

Figure 6: Lookup from higher to lower cluster.

A super node in this pattern executes the transformed process that creates a request message for iterative lookup from the request message for recursive lookup. Therefore, when the originator sends a request message for recursive lookup, lookup is executed at the super node of the destination cluster (Fig. 6 (1), (2)). The super node executes the transformed process, and forwards destination by using iterative lookup (Fig. 6 (3), (4)). The value is presented by using the communication shown in Fig. 6 (4). The super node sends a reply message including the value for recursive lookup to the originator (Fig. 6 (5)). Incidentally, the originator has to wait $2T_r$ because the lookup uses iterative lookup in the middle of lookup. By using iterative lookup at lower clusters where the churn rate is high, this pattern can shorten the latency of the entire lookup more than that with only recursive lookup.

5 EXPERIMENTS

5.1 Presupposition

We implemented the lookup in the Overlay Weaver[11] to evaluate our lookup strategy and compared its performance with that of only recursive or iterative lookup.

First, the setting for running the simulation and the version of the Overlay Weaver were:

- OS: Windows 7 Professional 64 bits
- CPU: Intel Core i5 3.2 GHz
- Memory: 4.0 GB
- Overlay Weaver: Ver. 0.10

Table 3 summarizes the parameters we set in the simulation.

Table 3: Parameters in simulation.

Number of nodes (N)	1000
Number clusters (C)	4
Latency of one hop (t)	6 ms
Recursive timeout (T_r)	84 ms
Iterative timeout (T_i)	15 ms

The number of nodes and clusters is defined by parameters in the work of Sato[10]. Sato's simulation used a Chord network by a minimum of 250 nodes. Although we set 250 nodes in a cluster, we have to discuss the number of clusters. C also means the number of super nodes, and C among N works as super nodes. Then, the lower-level cluster is built by other nodes as normal nodes. Therefore, a lower-level cluster has 250 nodes. Normal nodes have no relationship with the distribution of R, and there is not much difference between the numbers of nodes in each cluster. Thus, we assumed that latency t is 6 msec. This simplifies our simulation, so it does not consider a real environment. In actuality, t may be longer than this value.

Next, both timeout is set as a random value, so 84 and 15 ms mean maximum value. T_r is based on the definition expressed in Subsection 2.2. We assumed that path length l was defined as $O(\log N')$ when N' was N/C as the number of nodes in the lower-level clusters. Also, we considered that it had the lookup of top-level clusters and a potential of over $O(\log N')$, so we added various values to l. T_r is defined by multiplying t by l. Similarly, T_i is the value multiplying t by 2 and adding a slight allowance because a node has to wait for a response in iterative lookup. Therefore, T_r and T_i are defined formula (4), (5) when the various value is r. These timeouts are set for all nodes, but the originator does not wait for a timeout. If the originator uses iterative lookup, the lookup will soon end by a timeout. Thereby, originator has to wait for the response of the super node.

$$T_r = t\left[\log\left(\frac{N}{C}\right) + 1 + r\right] \quad (4)$$

$$T_i = 2t + r \quad (5)$$

Path length l is generally determined to be the key ID, which is a parameter that is not included in Table 3. This key ID is used the same as key ID to equalize the effect of l in all simulations as much as possible. By equalizing the effect, we ran the simulation for the key ID 100 times, and measured the average. In addition, we assumed that a higher and lower cluster were the same cluster in every simulation. We also assumed that churn rate p of higher clusters using recursive lookup was one at all times and p in lower clusters using iterative lookup could be set freely. According to formula (1), p means the churn rate and S needs to be nearly a fixed value. For this reason, nodes must not repair routing tables by churn during lookup. Additionally, the stabilization process was set to a large interval of 125 sec. This means $E[S]$ had a fixed value because nodes repaired fewer routing tables due to the stabilization process.

In addition, the following shows the routing tables of nodes.

- Predecessor node
- Successor List (no more than eight successor nodes)
- Finger table
- Normal nodes have super nodes in the cluster
- Super nodes have other super nodes in top-level cluster

When a normal node forwards a request message to another cluster, the node can forward the message to a super node in the same cluster in one hop. Additionally, a super node knows all other super nodes in the lookup for the top-level cluster and can forward the message to the super node of the destination cluster in one hop.

We considered lookup where a normal node forwards request messages to the node of another cluster. Additionally,

there are three lookup patterns for a cluster, and each lookup is executed in different nodes.

We measured latency from higher to lower clusters and otherwise with each lookup strategy using the above parameters.

5.2 Results

We measured average latency with simulation. Here, we assumed that the latency was the time until the destination node received a request message. In addition, the time also included the internal processing time of each node. Therefore, it measured $E[RL]$ and $E[IL]$ as follows in this simulation.

$$E[RL] = lt + \frac{1-p^l}{p^l} T_r \quad (4)$$

$$E[IL] = 2(l-1)t + \frac{1-p}{p} lT_i \quad (5)$$

First, we will consider pattern (B) in Subsection 4.3, which is a lookup whose destination cluster is higher. It assumes that the p of the higher cluster and that of the super node that belongs to a lower cluster are set to one at all times. Also, the originator does not leave the network. Additionally, we assumed that there was one lower cluster and three higher clusters. Therefore, we measured the average latency of nine lookup patterns that forward request messages to higher clusters. The results obtained from simulation are presented in Fig. 7.

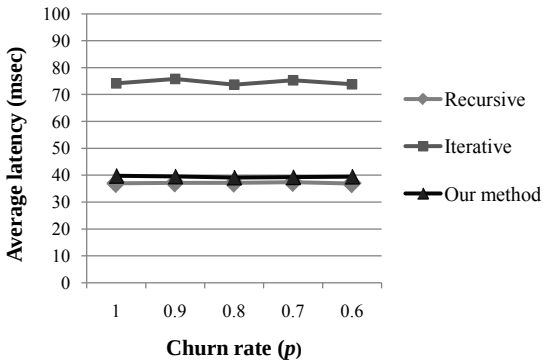

Figure 7: Average latency to three higher clusters by each lookup strategy.

This lookup pattern has little relevance to churn rate. The first address is the super node belonging at the cluster because this lookup pattern necessarily forwards a request message to other clusters. The super node forwards the request message to a super node belonging at the destination cluster. Each node assumes that churn does not occur. In addition, churn also does not occur after that because destination cluster is higher. As a result, the average latency barely changes at all under any p. In Fig. 7, when all nodes are steady state, the latency of our method is half that of iterative lookup. Our method has latency comparable to that of recursive lookup.

Second, we will consider pattern (A), which is the lookup from a higher to lower cluster. There are three lookup patterns from three other higher clusters. We set lower cluster to p, which is single value from 1 to 0.6. Also, we ran a simulation for each p 100 times and measured the average latency in each lookup strategy. The results are shown in Figure 8.

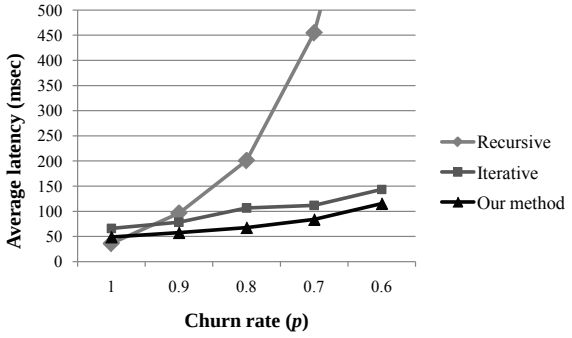

Figure 8: Average latency to one lower cluster by each lookup strategy.

If churn increases in Fig. 8, the average latency also increases. Recursive and iterative lookups are much the same as those in Fig. 1. However, our method performs the same as recursive lookup when p is one. If p decreases, increment of the average latency is similar to that of iterative lookup. Also, the results of our method are not identical to those of iterative lookup. Margin of average latency on each lookup is invariant from $p = 1$ to $p = 0.6$. Recursive lookup has the best average latency at only $p = 1$. However, from $p = 0.9$, recursive lookup has the worst average latency.

Here, we will think expected latency of this structure. This means the latency when any node forwards. Also, this has relevance to the structure. For example, the above simulation has one lower cluster and three higher clusters. If higher cluster is more than lower cluster, it is generally expected better latency. Because there is a high probability that the destination cluster is higher. On the other hand, if lower cluster is more than higher cluster, expected latency becomes low because it is a high probability that the destination cluster is lower.

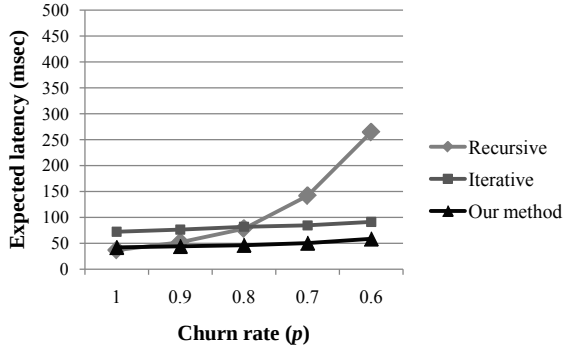

Figure 9: Expected latency on one lower cluster and three higher clusters.

For this reason, by using these results, we measured the average latency of the structure. This was measured by multiplying each of average latency when the destination cluster is both higher and lower by the number of higher or lower

clusters. In this case, results in Fig. 7 are multiplied by three as the number of higher clusters and that in Fig. 8 by one as the number of lower clusters. Then, it measured the average of these results. We assumed that it is expected latency on the structure. Figure 9 shows the results in the case of one lower cluster and three higher clusters.

This hierarchical DHT is made mostly of higher clusters, and so the expected latency is better than average latency to lower clusters. Iterative lookup and our method have flat latency as well. Also, recursive lookup has better average latency than average latency of only lookup to lower clusters.

Here, we think about the relationship between the average latency and the number of each cluster. In above case, we show the average latency when the structure is one lower cluster and three higher clusters. We think that the average latency is influenced by the number of lower and higher clusters. Therefore, we considered simulations that have different numbers of these clusters within *C*.

First, we ran a simulation in which the structure has two lower clusters and two higher clusters. Each lower cluster is set the same *p*. In this case, we obtained six lookup patterns in which the destination cluster is lower. Also, there are six lookup patterns that destination cluster is higher. As shown in Fig. 7 and 8, we measured the average latency in each lookup pattern. Figure 10 and 11 show each of average latency for lower and higher clusters.

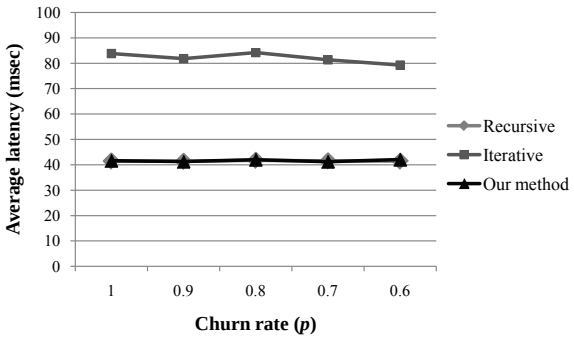

Figure 10: Average latency to two higher clusters by each lookup strategy.

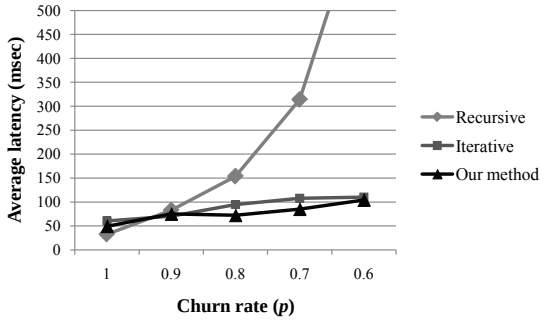

Figure 11: Average latency to two lower clusters by each lookup strategy.

These streams are not much more than Fig. 7 and Fig. 8. The result of Fig. 10 is a little higher than that of Fig. 7. Also, that of Fig. 11 becomes low a little. However, these results are evaluated relatively, and they mostly equal. We will discuss minor margin about their data on Section 6. Similarly, by these results, we measure expected latency of this structure. The result is shown Fig. 12.

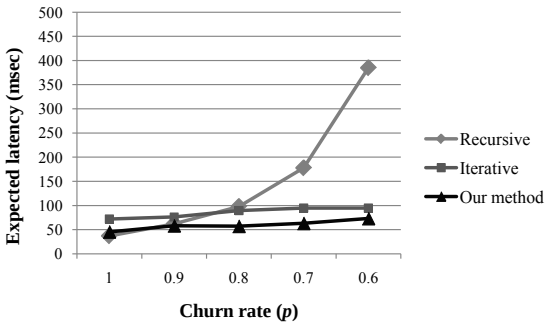

Figure 12: Expected latency on two lower clusters and two higher clusters.

This result is totally a little higher than result of Fig. 9. When *p* is 0.8, the result of Fig. 9 is that recursive lookup is lower than iterative lookup. However, Fig. 12 shows that recursive lookup is higher than iterative lookup under the churn rate.

Second, we ran simulation that structure has three lower clusters and one higher cluster. In this case, lookup patterns that destination cluster is higher are three patterns. There are nine lookup patterns that destination cluster is lower. We measured the average latency each lookup pattern similarly. The average latency of the pattern that destination cluster is higher is shown as Fig. 13. Also, we show the average latency to lower clusters in Fig. 14.

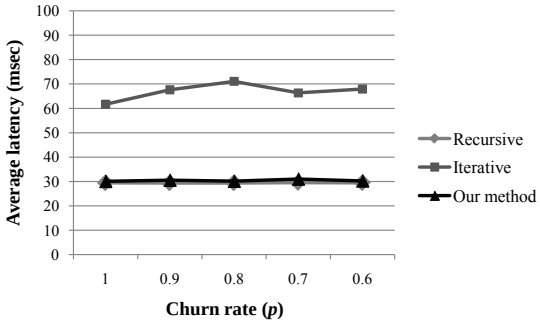

Figure 13: Average latency to one higher cluster by each lookup strategy.

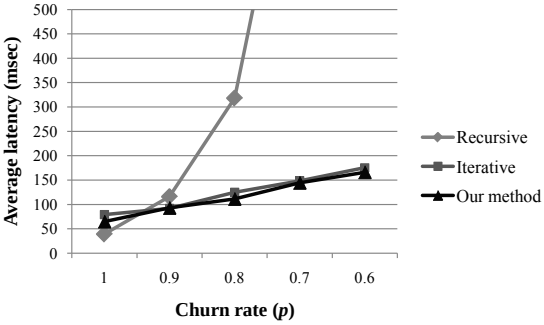

Figure 14: Average latency to three lower clusters by each lookup strategy.

These results have mostly the same stream. However, max value of average latency to lower clusters is higher than other results to lower cluster. On the other hand, max value of average latency to higher cluster is better than other results. Similarly, by these results, we measure expected latency of this structure, and the result is shown Fig. 15.

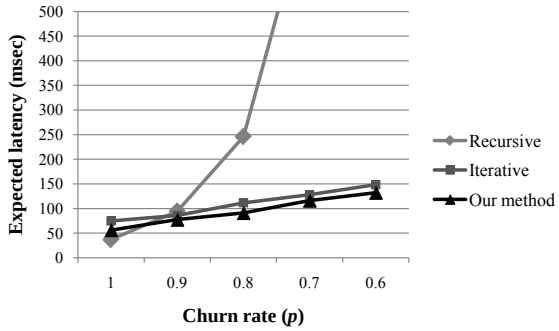

Figure 15: Expected latency on three lower clusters and one higher cluster.

These results are much higher than other results of expected latency. However, the stream of the results is not much more than those of other results. In addition, when p is 0.9, the expected latency of recursive lookup is a little higher than other lookup strategies.

According to these results, when nodes forward request messages to higher clusters, our method provides the performance of recursive lookup. Also, our method provided the performance of iterative lookup when nodes forward request messages to lower clusters. This is possible under any churn rate at lower clusters and proportions of higher to lower clusters. As a result, our method is effective when compared with only recursive or iterative lookup under any state and structure.

6 DISCUSSION

We will discuss the above results. First, we note the effect of our method. When the destination cluster is higher, our method performs similarly to recursive lookup under any churn rate and structures. Also, when destination cluster is lower, our method performs similarly to performance of iterative lookup under any situations. As a result, the expected latency of our method is relatively better than other lookup strategies as integrated evaluation.

Second, we note the average latency of lookup to higher cluster on each structure. For the average latency of each lookup strategy, although the rate is almost the same, the max value each of average latency is much different. This is considering that all lookup patterns have different path lengths. The path length is five at minimum and eleven at maximum. If the path is long, latency becomes high. Therefore, average latency becomes high in the patterns that have long path length. For this reason, the results in Fig. 10 include patterns that have long path length, and those in Fig. 13 do not. However, we can find the integrate effect of path length by measuring expected latency. If we consider the effect of churn rate, path length may have to be fixed.

In the results of expected latency, when higher cluster is defined as $p = 0$, if super nodes know the churn rate of each cluster and a number of clusters, we can evaluate effectively the lookup strategy under the churn rate. For example, if p of a cluster becomes 0.8, the cluster uses iterative lookup when there are already two clusters using recursive lookup and one cluster using iterative lookup. This is shown in Fig. 12.

However, the case in which p is 0 is less common in P2P. For this reason, we have to define higher and lower clusters. Therefore, we will research rigorous p, the structure of clusters, and the effects of different numbers of nodes and the clusters.

Finally, we attended to recursive + ACK lookup strategy. This lookup strategy has best expected latency among lookup strategies under any churn rate. $E[RL+ACK]$ is defined in the following, and Fig. 16 shows comparison of $E[RL+ACK]$ with $E[RL]$ and $E[IL]$ under different p.

$$E[RL + ACK] = (l+1)t + \frac{1-p}{p} lT \qquad (6)$$

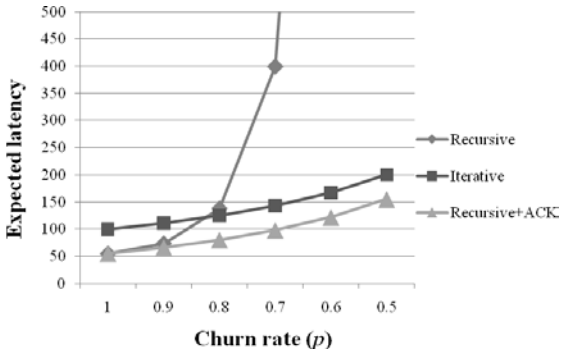

Figure 16: Expected latencies of recursive, iterative, and recursive + ACK lookups under different p.

Here, we had a question, why do not we use recursive + ACK in hierarchical DHT. Thereby we researched about advantage of iterative lookup. According to Ref.[12], iterative lookup has some advantages that recursive lookup lacks, fate-sharing, debugging, compartmentalization, and route table extraction. We think that these advantages are effective on mobile P2P environment where a lot of nodes abruptly disconnect P2P network. Because intermediate peers of mobile have probability of lost messages. However, for successful forwarding, it is important that high reliability nodes use iterative lookup, because it has fate-sharing. In our proposed method, super nodes use iterative lookup at lower clusters. So we think that our method is more reliable forwarding than recursive + ACK lookup strategy.

7 CONCLUSION

We noted the effect of churn for recursive and iterative lookups in this study, and there were differences in the churn rate for each cluster on hierarchical DHT when the reliability of nodes was considered. We proposed a lookup method that will leverage both lookup advantages by selecting relevant lookup strategy at each cluster. Additionally, we demonstrated that the new approach is significantly bet-

ter in comparison to only recursive or iterative lookups. As a result, our method had the best expected latency under any churn rate. In future work, we need to consider an approach that dynamically applies our method to a DHT system. Additionally, we intend to propose an adaptive method that is able to adjust to variations in clusters by specifically defining the reliability of nodes and measuring the churn system. Also, we intend to consider various other parameters for the lookup strategy and how to provide optimal lookup.

REFERENCES

[1] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek and H. Balakrishnan, ``Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,'' Proceedings of SIGCOMM'01, pp.149-160 (2001).

[2] P. Maymounkov and D. Mazieres, ``Kademlia: A Peer-to-peer Information System Based on the XOR Metric,'' Proceeding of IPTPS'01, pp.53-65 (2002).

[3] A, Rowstron and P. Druschel, ``Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,'' Proceedings of Middleware 2001, pp.329-350 (2001).

[4] K. Shudo, D. Kato, Y. Kadobayashi and Y. Doi, ``A Comparative Study of Iterative and Recursive Lookup Styles on Structured Overlays,'' IPSJ SIG Technical Reports, Vol.2006, No.86(OS-103), pp.9-16, (2006). (*in Japanese*)

[5] S. Doi, S. Matsuura, K. Fujikawa and H. Sunahara, ``Churn Tolerant Overlay Network Using Time Layered and Time Aggregation Methods,'' IPSJ Journal, Vol.51, No.4, pp.1142-1151 (2010). (*in Japanese*)

[6] S. Saroiu, ``Measurement and analysis of Internet content delivery systems,'' Doctoral Dissertation (2004).

[7] D. Wu, Y. Tian and K. W. Ng, ``An analytical study on optimizing the lookup performance of distributed hash table systems under churn,'' Concurrency and Computation: Practice & Experience, Vol.19, No.4, pp.543-569 (2007).

[8] L. Garces-Erice, E. W. Biersackm, P. A. Felber, K. W. Ross and G. Urvoy-Keller, ``Hierarchical Peer-to-peer Systems,'' Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing(Euro-Par), pp.1230-1239 (2003).

[9] S. Zoels, Z. Despotovic and W. Kellerer, ``On hierarchical DHT system - An analytical approach for optimal designs,'' Computer Communications, Vol.31, No.3, pp.576-590 (2008).

[10] F. Sato, ``Configuration Method for Hierarchical DHT Systems Based on Join/Leave Ratio,'' IPSJ Journal, Vol.51, No.2, pp.418-428 (2010). (*in Japanese*)

[11] K. Shudo, ``Overlay Weaver: An Overlay Construction Toolkit,'' HTML available at, http://overlayweaver.sourceforge.net/index-j.html.

[12] D. Stutzbach and R, Rejaje, ``Improving Lookup Performance over a Widely-Deployed DHT,'' Proceedings of INFOCOM 2006, pp.1-12 (2006).

(Received February 27, 2012)
(Revised November 23, 2012)

Tomonori Funahashi received his B.E. and M.E degrees in information science from Future University Hakodate, Japan in 2011 and 2013. His research interests include structured P2P network and protocol, heterogeneous P2P on fixed and mobile environment.

Yoshitaka Nakamura received B.E., M.S., and Ph.D. degrees from Osaka University in 2002, 2004 and 2007, respectively. He is currently a research associate at the School of Systems Information Science, Future University Hakodate. He is a member of IEEE and IPSJ.

Yoh Shiraishi received doctor's degree from Keio University in 2004. He is currently an associate professor at the Department of Media Architecture, School of Systems Information Science, Future University Hakodate Japan. His research interests include database, mobile sensing and ubiquitous computing. He is a member of IPSJ, IEICE, GISA, and ACM.

Osamu Takahashi received master's degree from Hokkaido University in 1975. He is currently a professor at the Department of System Information Science at Future University Hakodate, Japan. His research interest includes ad-hoc network, network security, and mobile computing. He is a member of IEEE, IEICE, and IPSJ.