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 that nodes leave the network. When churn occurs infrequently, recursive lookup outperforms iterative lookup, but it returns back when churn occurs frequently. 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 t establish the neighboring churn rate. Hierarchical DHT takes into consideration the reliability of nodes to ascertain the churn rate. Therefore, we compared our lookup strategy with the use of either recursive or 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 take into consideration 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. There is hierarchical DHT with advanced features that has taken into consideration how reliability of node is [10]. This has a clustering method that establishes the reliability of nodes. Thus, each cluster is established the reliability approximately.

We propose applying an optimal lookup strategy to each cluster on hierarchical DHT that takes into consideration the reliability of nodes and separates recursive and iterative lookups efficiently in this paper.

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 successor list which has a number of successor nodes, and a finger table that can route efficiently to the routing table. Chord completes path length $O(\log N)$ with these routing tables when N is the number of 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 strategy that originator node which demands value requests lookup other nodes. However, iterative lookup is a method which 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 the lookup has 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 received a request message does not have the purposed value, it forwards the request message to a node that is closer to the destination than itself. This process is executed till the request message reaches the destination node (Figure 1 (2), (3)). In contrast, the originator receives reply messages for request messages after the message has been forwarded in iterative lookup (Figure 1 (1-2), (2-2)).

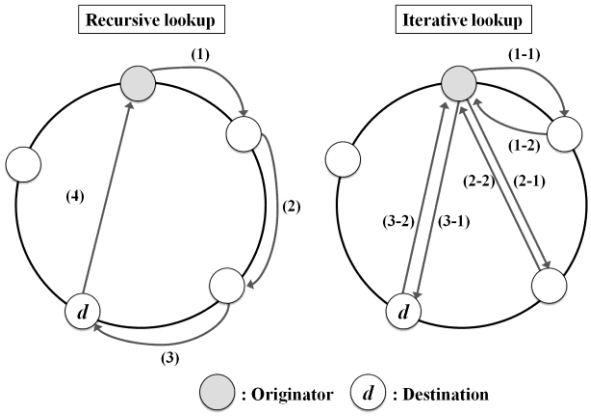

Figure 1: Recursive and iterative lookup strategy on Chord when path length = 3.

When a node received request message does not have the purposed value, it makes the reply message including the addresses of nodes which are closer to the destination than itself. The originator node forwards a request message to the destination node by using the address included in the reply message.

The performances of recursive and iterative lookups are affected by these communication methods and churn where nodes leave the network. The system churn rate, which is the probability what nodes will leave the network, is determined 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 shows 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].

$$p = \frac{E[R]}{E[R] + E[S]} \qquad (1)$$

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 occur in neighbor nodes that have shorter lifetimes. That is, the churn rate is low when p is high, and it is high when p is low. More specifically, p means the churn rate in the network when $E[S]$ approximates a fixed value.

The performance of recursive and iterative lookups are defined by using churn rate p and latency of communication [7]. 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 message 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 $T_i \geq 2lt$. As a result, the expected latency of recursive lookup $E[RL]$ is defined in the following by these parameters.

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

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

$$E[IL] = 2lt + \frac{1-p}{p} lT_i \qquad (3)$$

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.

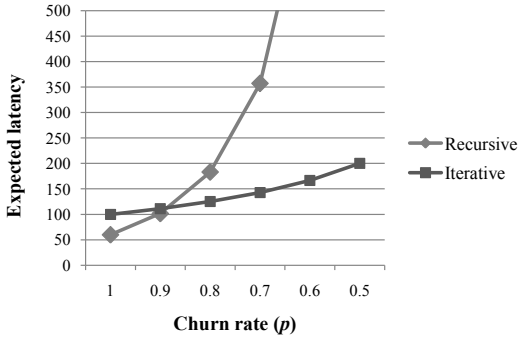

Figure 2: Expected latencies of recursive and iterative lookups under different p.

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 increase out the performance of recursive and iterative lookups, we need determine what the system churn rate is.

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 specific a super node belong to lower-level cluster. The super node provides normal nodes with routes to other clusters.

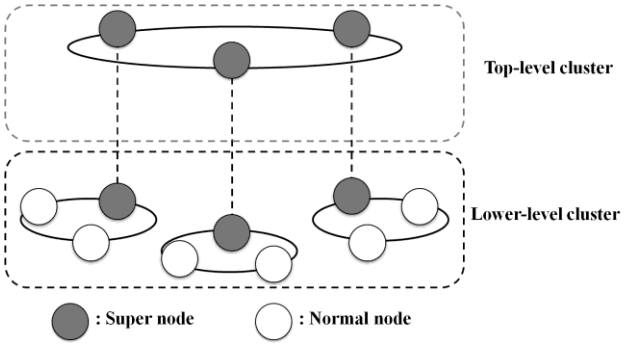

Figure 3: Example of two-tier hierarchical DHT.

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 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 as to 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 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 takes reliability into consideration. 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 PROBLEM

When p is low in recursive and iterative lookups, recursive lookup has an advantage, but iterative lookup has an advantage when p is high. We culled the lookups 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 considered applying each lookup to that. However, each message format in recursive and iterative lookups uses differences for that. For this reason, a lookup cannot contact another lookup.

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 take reliability into consideration. We used the Chord algorithm because it had various features, e.g., it had a simple structure and was scalable. We also made note of the stabilization process for the reason of 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 to 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 about 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 to determine what value is received for which request message, and the Time To Live (TTL) which is set infinitely to forward request messages. The Identifier is set like the time made the request message. 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 Figure 4.

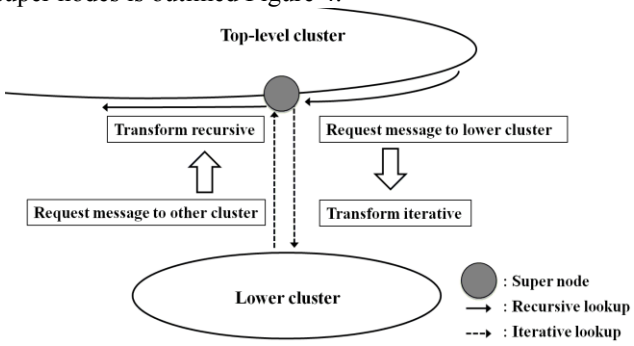

Figure 4: Transformed process at super node of Lower cluster.

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, and TTL. For this reason, the super node creates a new identifier for the request message by the time made the 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 it is sent the value of the transformed format.

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, Figure 5 shows an example of pattern (A).

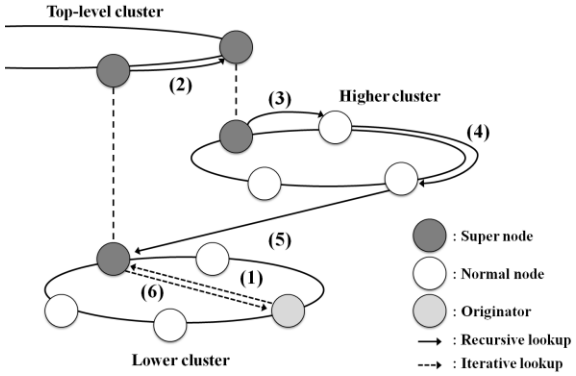

Figure 5: Lookup from lower to higher cluster.

The flow for lookup where request and reply messages are forwarded is indicated by the number in Figure 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 (Figure 5 (1)). The super node transforms the message at the start, and starts recursive lookup. The lookup forwards to super and destination nodes (Figure 5 (2)-(4)). Although the destination does not directly send the value to the originator, it sends the super node belonging to the originator (Figure 5 (5)). The super node transforms the received message, and sends data to the originator (Figure 5 (6)). In this case, originator node waits for the message as Figure (6). However, the time may exceed the timeout of Iterative lookup. Here, we assume that super nodes do not leave the network, so some nodes certainly can communicate to super nodes. For the assumption, originator node waits to receive reply message from super node, because the node makes a reply certainly.

We consider that this pattern shorter 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, Figure 6 shows an example of pattern (B).

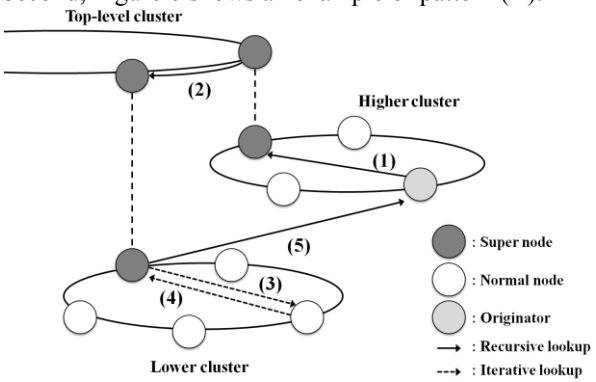

Figure 6: Lookup from higher to lower cluster.

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 (Figure 6 (1), (2)). The super node executes the transformed process, and forwards destination by using iterative lookup (Figure 6 (3), (4)). The value is presented by using the communication shown in Figure 6 (4). The super node sends a reply message including the value for recursive lookup to the originator (Figure 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.

No. of nodes (N)	1000
No. of clusters (C)	4
Latency of one hop (t)	6 msec
Recursive timeout (T_r)	84 msec
Iterative timeout (T_i)	15 msec

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. Normal nodes have no relation to the distribution of R, and there is not much difference between the numbers of nodes in each cluster. 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 one of the lower-level clusters. Also, we considered that it had the lookup of top-level clusters and a potential of over $O(\log N')$, and we added various values to l. T_r is defined by multiplying t by l. Similarly, T_i is multiplying t by 2 and adding a slight allowance because a node has to wait for a response in iterative lookup. 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 needs S which is nearly a fixed value. For this reason, nodes repair fewer routing tables by churning during lookup. Additionally, the stabilization process was set to a large interval of 125 msec. 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 (not 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 of other super nodes in the lookup for the top-level cluster, and can forward the message to 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 is 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 Figure 7.

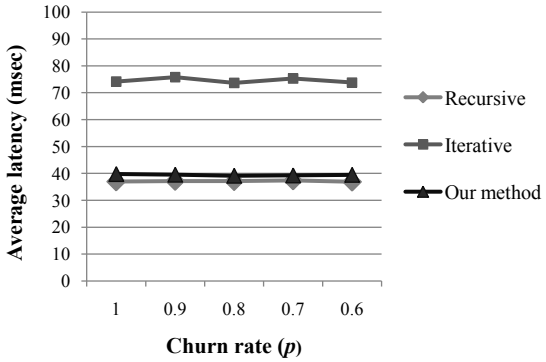

Figure 7: Average latency to three higher clusters by each lookup strategy.

This lookup pattern has little relevance to churn rate. First address is the super node belonging at the cluster because this lookup pattern necessarily forwards a request message to other cluster. The super node forwards the request message to super node belonging at destination cluster. Each node is assumed 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 hardly changes at all under any p. In Figure 7, when all nodes are steady state, the latency of our method is twice as short as that of iterative lookup. Although it is compared to recursive lookup, it has nearly latency of that.

Second, we will consider pattern (A) which is the lookup from higher to lower cluster. There are three lookup patterns from other three 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 result shows in Figure 8.

If churn increases in Figure 8, the average latency also increases. Recursive and iterative lookups are much the same as Figure 1. However, our method is same well as recursive lookup when p is one. If p decreases, increment of the average latency is similar to that of iterative lookup. Also, the result of our method is not identical with that 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.

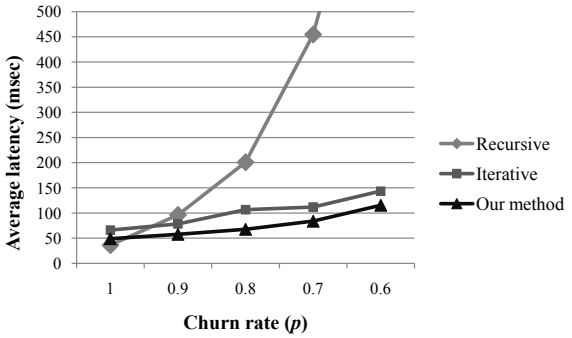

Figure 8: Average latency to one lower cluster by each lookup strategy.

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, there are one lower cluster and three higher clusters in above simulation. If higher cluster is more than lower cluster, it is generally expected better latency. Because it is high probability that destination cluster is higher. On the other hand, if lower cluster is more than higher cluster, expected latency becomes low because it is high probability that destination cluster is lower.

For this reason, by these results, we measured the average latency of the structure. This was measured by multiplying each of average latency which destination cluster is both higher and lower by the number of higher or lower clusters. In this case, it multiplies result of Figure 7 by three as the number of higher clusters and that of Figure 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 result of the case that there are one lower cluster and three higher clusters.

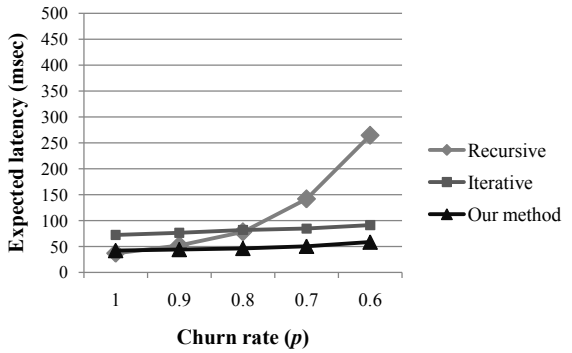

Figure 9: Expected latency on one lower cluster and three higher clusters.

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

Here, we think about relationship between the average latency and the number of each cluster. In above case, we show the average latency that 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 which have different the number of these clusters within *C*.

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

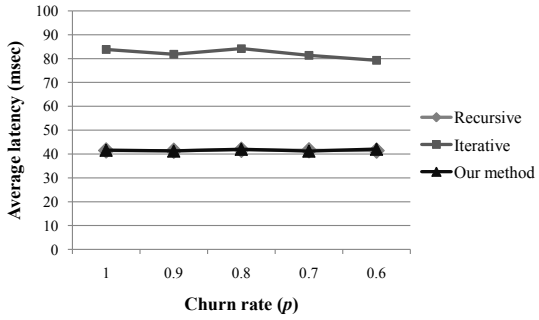

Figure 10: Average latency to two higher clusters by each lookup strategy.

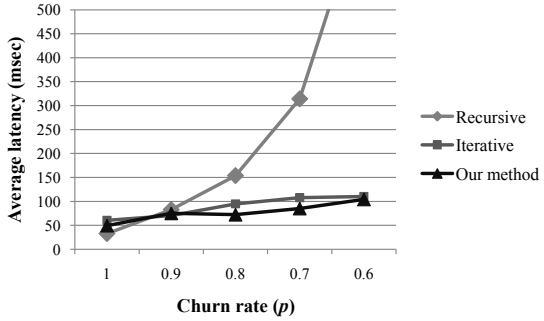

Figure 11: Average latency to two lower clusters by each lookup strategy.

These streams are not much more than Figure 7 and Figure 8. The result of Figure 10 is a little higher than that of Figure 7. Also, that of Figure 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 Figure 11.

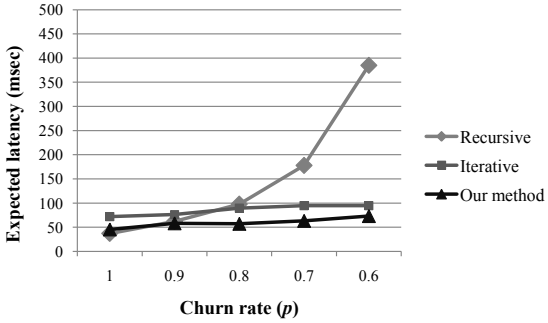

Figure 11: Expected latency on two lower clusters and two higher clusters.

This result is totally a little higher than result of Figure 9. When *p* is 0.8, the result of Figure 9 is that recursive lookup is lower than iterative lookup. However, Figure 11 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 Figure 12. Also, we show the average latency to lower clusters in Figure 13.

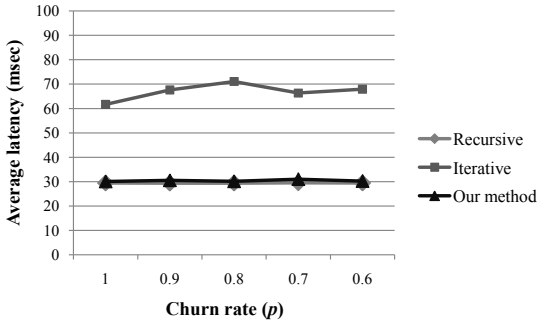

Figure 12: Average latency to one higher cluster by each lookup strategy.

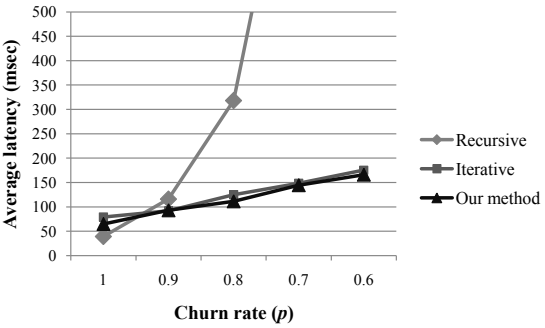

Figure 13: Average latency to three lower clusters by each lookup strategy.

These results have mostly 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 Figure 14.

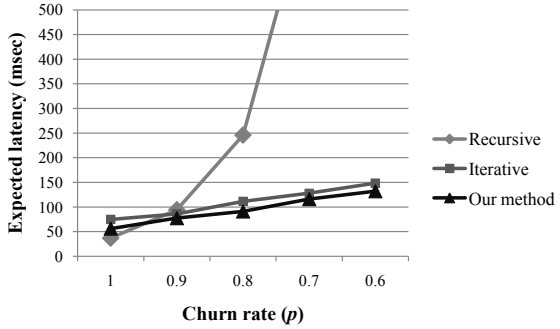

Figure 14: Expected latency on three lower clusters and one higher cluster.

This result is much higher than other results of expected latency. However, stream of the result is not much more than other results. In addition, when p is 0.9, the expected latency of recursive lookup is a little higher than other lookup strategy.

By these results, when nodes forward request messages to higher cluster, our method provides the performance of recursive lookup. Also, our method provided the performance of iterative lookup when nodes forward request messages to lower cluster. This is possible under any churn rate at lower cluster and proportion of higher to lower cluster. 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 about above results. First, we note effect of our method. In the case which destination cluster is higher, it has the similar performance of recursive lookup under any churn rate and structures. Also, when destination cluster is lower, it has the similar performance of iterative lookup under any situations. As a result, the expected latency of our method is relatively better than only other lookup strategy 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 same, the max value each of average latency is much different. This is considered that each lookup pattern have different path length. The path length is that it is five at minimum and eleven at maximum. If path length is long, latency becomes high. Therefore, average latency included the pattern had long path length becomes high. For this reason, the result of Figure 10 includes patterns had long path length, and that of Figure 12 does not include the patterns. However, the patterns also have reference to another lookup. For example, the result of Figure 13 becomes much high because it includes the patterns had long path length. However, we think that integrate effect of path length by measuring expected latency. If we consider effect of churn rate definitely, path length may have to be fixed.

For the results of expected latency, when higher cluster is defined $p = 0$, if super nodes know churn rate of each cluster and a number of clusters, we think that evaluate effective 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 cluster using recursive lookup and one cluster using iterative lookup. This can know by Figure 11.

However, the case that p becomes 0 is less common in P2P. For this reason, we have to define higher and lower cluster. Therefore, we have to research about rigorous p and structure of clusters, a number of nodes and clusters.

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 culling the lookup strategy for each cluster. Additionally, we demonstrated that the new approach is significant 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 the SIGCOMM, 2001.

[2] P.Maymounkov, D.Mazieres, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric”, IPTPS02, pp.53-65, 2002.

[3] A,Rowstron and P.Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp.329-350, 2001.

[4] K.Shudo, D.Kato, Y.Kadobayashi, Y.Doi, “A Comparative Study of Iterative and Recursive Lookup Styles on Structured Overlays”, The Special Interest Group Notes of IPSJ “system software and Operating System”, Vol.86, pp.9-16, 2006.

[5] S.Doi, S.Matsuura, K.Fujikawa, H.Sunahara, “Churn Tolerant Overlay Network Using Time Layered and Time Aggregation Methods”, Transactions of Information Processing Society of Japan Vol. 51, No.4, pp.1142-1151, 2010.

[6] S.Saroiu, “Measurement and analysis of Internet content delivery systems.”, Doctoral Dissertation, 2004.

[7] D.Wu, Y.Tian,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, 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”, ACM/IFIP International Coference on Parallel and Distributed Computing, 2003.

[9] S.Zoels, Z.Despotovic, W.Kellerer, “On hierarchical DHT system - An analytical approach for optimal designs”, Computer Communications, Vol.31, pp.576-590, 2008.

[10] F.Sato, “Configuration Method for Hierarchical DHT Systems Based on Join/Leave Ratio”, Transactions of Information Processing Society of Japan Vol.51, No.2, pp.418-428, 2010.

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