FRT-2-Chord: A DHT Supporting Seamless Transition between One-hop and Multi-hop Lookups with Symmetric Routing Table

Yasuhiro Ando, Hiroya Nagao, Takehiro Miyao and Kazuyuki Shudo Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552 Japan
Email: {yasuhiro.ando, hiroya.nagao, takehiro.miyao, shudo}@is.titech.ac.jp

***Abstract*—FRT-2-Chord is a DHT based on Flexible Routing Tables, a method for designing routing algorithms for overlays. DHTs should consider the following factors when constructing overlay topologies: the number of nodes, identifier distance, network proximity and node groups. Existing DHTs either do not adapt to those factors or focus on only some of them. FRT-2-Chord is more adaptable and achieves efficient routing under a variety of conditions of those factors. Theoretical analysis and experimental results show the efficiency of FRT-2-Chord.**

***Keywords*—overlay network, structured overlay, peer-to-peer, DHTs, routing**

I. Introduction

Overlay networks offer scalability and fault tolerance for peer-to-peer systems.They are classified into two types: unstructured and structured. In unstructured overlay networks, requests are flooded or randomly routed to find data items. In structured overlay networks, the overlay topology is organized based on constraints and a request is routed efficiently to data items by a set of logical rules. Structured overlays are used primarily by distributed hash tables (DHTs).

Many peer-to-peer systems impose requirements such as low latency, low bandwidth utilization and lookup correctness. To satisfy these requirements, DHTs should consider the following factors when constructing their overlay topologies: the number of nodes, identifier distance, network proximity, node groups and so on. If a DHT could not adapt to changes in these factors, it could not offer scalability and fault tolerance. At a minimum, DHTs should meet the following requirements for adaptability.

1) **Routing efficiency independent of the number of nodes.** In peer-to-peer systems, the number of nodes and the rate of joins and leaves change dynamically and depend on applications.
   1-a) **Short path length.** When the number of nodes is larger than the maximum number of neighbors per node, a query reaches its destination node in multiple hops. In a DHT, a query is routed in $O(\log N)$-hops by using only $O(\log N)$ neighbors per node, where N is the number of nodes. As an example of the importance of this point, the number of nodes in the structured overlay network which is constructed by BitTorrent [1] [2] clients reaches 10 millions [3].
   1-b) **Ability to route a query in one-hop.** If the number of nodes is less than or equal to the size of the routing table, an node should forward a query to its destination directly, that is, in a *one-hop* lookup. For example, distributed storage systems such as Amazon Dynamo [4] and Apache Cassandra [5] [6] use a DHT to provides one-hop lookups. Using such a DHT, responsibility for data items is distributed among the nodes without centralized control and data access latency is reduced.
2) **Handling of factors that affect neighbor selection.** A DHT should take account of network proximity and node groups in addition to short path length. If network proximity is ignored, a lookup may be routed through nodes that are far away in the physical network. For example, proximity-aware routing tables enable file sharing systems to reduce data access latency. When node groups is not considered, hops across Internet service providers or data centers may occur frequently. For example, distributed storage systems account for node groups in the same rack or in the same datacenter when placing and accessing replicas [5] [6] [7]. A DHT should consider those independent factors when constructing routing tables.
   2-a) **Flexibility of a routing table construction.** A DHT is designed to consider identifier distance when constructing routing tables. Extensions to a DHT may consider other factors. Each extension focuses on a single factor because the factors are independent. Hence, flexibility of a routing table construction is required for a DHT to be able to consider each of these factors.
   2-b) **Symmetry of routing tables.** *Symmetry of routing tables* is the following property: when the routing table of node x contains node y, the routing table of node y tends to contain node x. Symmetry of routing tables reinforces consideration to network proximity and node groups. The consideration takes effect on the nodes in a pair because those factors are symmetric between them. Moreover, symmetry of routing tables reduces routing table maintenance

This work was supported by MEXT KAKENHI (24650025 and 25700008).

TABLE I. DHTs AND REQUIREMENTS.

	(1-a)	(1-b)	(2-a)	(2-b)
Pastry	S	N	N	S
Chord	S	N	N	N
Kademlia	S	N	N	S
S-Chord	S	N	N	S
EpiChord	N	N	N	-
OneHop	N	S	N	-
FRT-Chord	S	N	S	N
FRT-2-Chord	S	S	S	S

(1-a) : Short path length
(1-b) : Ability to route a query in one hop
(2-a) : Flexibility of routing table construction
(2-b) : Symmetry of routing tables
S : A DHT satisfies a requirement
N : A DHT does not satisfy a requirement

costs, because query messages also serve as heartbeat messages. That is, a node sends a message to another node in its routing table and the receiving node knows the sending node is alive.

Numerous DHTs have been proposed over the last decade [8] [9] [10] [11] [12] [13] [14]. However, existing DHTs do not satisfy all of above requirements or focus on only a part of them; this lack restricts the situations in which those DHTs offer efficient routing. This paper presents *FRT-2-Chord*, a DHT based on Flexible Routing Tables (FRT). FRT [14] is a method for designing routing algorithms. FRT-2-Chord satisfies all of the requirements detailed above.

II. RELATED WORK

Table I lists existing DHTs and FRT-2-Chord and shows whether each satisfies the requirements described in Section I. Chord [8], Pastry [9], S-Chord [11] and Kademlia [10] achieve $O(\log N)$-hop lookup with $O(\log N)$ neighbors per node. However, in these DHTs a routing table cannot contain all nodes in an overlay even if the routing table size is larger than N. It is due to such constraints imposed by these algorithms that a part of a routing table can hold only limited node identifiers. For example, neither a finger in Chord nor a k-bucket in Kademlia can hold arbitrary node identifiers. This restriction prevents queries from being routed in one hop. In EpiChord [12] and OneHop [13], the routing table size is large enough to contain all nodes in an overlay. However, to achieve this, EpiChord and OneHop do not work when the number of nodes is larger than expected. In FRT-Chord [14], an FRT-based DHT, a routing table can contain all nodes in an overlay as long as routing table size is larger than N. However, in Chord-based DHTs such as FRT-Chord and EpiChord, a query is forwarded to the destination node after being forwarded to its predecessor. These queries require two or more hops to reach the destination node; that is, those Chord-based DHTs do not achieve one-hop even when the routing table of the node that issued the query holds the destination node.

In OneHop and EpiChord, routing tables are naturally symmetry because they contain all nodes in the overlay. In Chord and FRT-Chord, routing tables are not symmetic due to their identifier distance described in Section II-A. In Pastry, Kademlia and S-Chord, routing tables are symmetic.

A. Chord

Chord is a DHT, in which identifiers are represented as a circle of numbers from 0 to $2^m - 1$. A node identifier is m bits long and chosen by hashing the node's IP address.

In a DHT, data are stored as key-value pairs. A data identifier is produced by hashing the key and the data is stored at a node according to the data identifier. In Chord, a node is responsible for a key if the node's identifier is equal to the identifier of the key or follows it. A query is routed according to the identifier distance from ID x to ID y, $\mathrm{d}(x, y)$,which is defined as follows.

Definition 1:

$$\mathrm{d}(x,y) = \begin{cases} y - x, & (x < y) \\ y - x + 2^m, & (y \leq x) \end{cases} \quad (1)$$

In Chord, each node maintains three routing tables (successor list, predecessor, and finger table). Each entry in a routing table is a node identifier and IP address pair. A successor list at node s contains a certain number of the closest nodes from s in the clockwise direction, and the successor is the closest node of those nodes. A predecessor contains the closest node from s in the anticlockwise direction. A stabilization protocol is used to guarantee reachability by maintaining the successor of each node.

The ith entry in the finger table is the first node that succeeds s by at least $2^{i-1} (i = 1, 2, ..., m)$ in the clockwise direction. In Chord, a query with the target identifier t is forwarded as follows.

1) Current node c forwards to the entry node e from which the distance to t is minimal in c's finger table. By repeating this forwarding, the query reaches the predecessor of t.
2) The predecessor of t forwards to the successor.

Chord achieves $\mathrm{O}(\log N)$-hop lookup performance with N nodes.

B. FRT-Chord

FRT is a method of designing routing algorithms for structured overlays. An FRT-based algorithm defines a total order on node identifier combinations in a routing table and then, repeatedly refines routing tables on the basis of the order.

FRT-Chord is an FRT-based DHT. In FRT-Chord, the identifier space and forwarding mechanism are the same as in Chord and the node responsible for a key is determined as in Chord. Each node maintains a single routing table E without distinguishing among a successor list, a predecessor, and a finger table, and FRT-Chord can dynamically change the size of the routing table. A routing table E at a node s is a set of entries $\{e_i\}$ which are ordered clockwise from s. Each entry e_i consists of a node identifier $e_i.\mathrm{id}$ and an IP address $e_i.\mathrm{address}$ (note that e_i is referred to as $e_i.\mathrm{id}$). By this definition e_1 and $e_{|E|}$ correspond to a successor and a predecessor.

1) Total Order $\leq_{\mathrm{ID}}$ of the Routing Table Set: FRT-Chord defines a total order $\leq_{\mathrm{ID}}$ on node identifier combinations in a routing table. Routing tables are refined according to the total order $\leq_{\mathrm{ID}}$. In FRT-Chord, the *reduction ratio* of a query forward is defined as $\mathrm{d}(E.\mathrm{forward}(t), t)/\mathrm{d}(s, t)$, where $E.\mathrm{forward}(t)$ is the entry in the routing table E, to which the query for target identifier t is forwarded by node s. A reduction ratio expresses how close to t the entry $E.\mathrm{forward}(t)$ is in comparison to s. In FRT-Chord, the reduction ratio of a forwarding to e_i takes its worst-case value when $t = e_{i+1}$. The *worst-case reduction ratio* $r_i(E)$ of forwarding to a node e_i for defining a total order $\leq_{\mathrm{ID}}$ is defined as follows:

Definition 2:

$$r_i(E) = \frac{\mathrm{d}(e_i, e_{i+1})}{\mathrm{d}(s, e_{i+1})} \ (i = 1, 2, ..., |E| - 1) \tag{2}$$

Let $\{r_{(i)}(E)\}$ be the list of redecution ratios arranged in descending order. The total order of the routing table set based on $\{r_{(i)}(E)\}$ is defined as follows.

Definition 3:

$$E \leq_{\mathrm{ID}} F \Leftrightarrow \{r_{(i)}(E)\} \leq_{\mathrm{dic}} \{r_{(i)}(F)\} \tag{3}$$

where $\leq_{\mathrm{dic}}$ is the lexicographical order.

2) Guarantee of Reachability: Any operations that guarantees reachability is called a *guarantee of reachability*. The stabilization protocol such as Chord realizes the guarantee of reachability in FRT-Chord. Entries concerning a guarantee of reachability are called *sticky entries*. For example, a successor list and a predecessor are sticky entries.

3) Entry Learning: In FRT, learning a node identifier and IP address and then inserting them into a routing table are called *entry learning*. Whenever one node is informed about another node's identifier and IP address, the node learns them.

4) Entry Filtering: When the number of entries $|E|$ exceeds the size of a routing table L, FRT-Chord will remove one of the entry from the current routing table in order to keep $|E| \leq L$ according to the order $\leq_{\mathrm{ID}}$. This entry removal operation is called *entry filtering*.

C. GFRT-Chord

GFRT-Chord [14] is an extension of FRT-Chord in order to consider node groups. It reduces hops across different node groups while keeping path lengths short. Its methods of guarantee of reachability and entry learning are the same as FRT-Chord. It is characterized by its entry filtering.

III. FRT-2-Chord

FRT-2-Chord is an FRT-based DHT. In FRT-2-Chord, the identifier space is the same as in Chord and entry learning is the same as in FRT-Chord.

A. Node Responsible for a Key and Forwarding

In FRT-2-Chord, the identifier distance from x to y, $\mathrm{d}(x, y)$, is defined as follows.

Definition 4:

$$\mathrm{d}(x, y) = \min\{|x - y|, 2^m - |x - y|\} \tag{4}$$

By this definition, $\mathrm{d}(x, y) = \mathrm{d}(y, x)$ holds for any x and y; that is, distance is symmetric. Since routing tables are constructed according to a total order on identifier distance in FRT-2-Chord, routing tables will also be symmetric. In FRT-2-Chord, the node responsible for a target identifier t is the node s that minimizes $\mathrm{d}(t, s)$. A node forwards a message to an entry node e in the routing table, where $\mathrm{d}(e, t)$ is minimized among any other entries. When there exists entries e, e' satisfying $\mathrm{d}(e, t) = \mathrm{d}(e', t)$ and $\mathrm{d}(e, t)$ is minimized among any other entries, a node forwards a message to e, where $\mathrm{d}_{\mathrm{chord}}(e, t) < \mathrm{d}_{\mathrm{chord}}(e', t)$, letting $\mathrm{d}_{\mathrm{chord}}(x, y)$ be the identifier distance in Chord. The node that terminates forwarding is the node responsible for t. In contrast to other Chord-based DHTs, in FRT-2-Chord the node responsible for a key is the destination node by definition of identifier distance. Therefore, a node is able to route a query in one hop, if its routing table contains the node responsible for t. In FRT-2-Chord, a node may forward a message in either the clockwise direction or anticlockwise direction. We call this the *two directions property*.

B. Total Order $\leq_{\mathrm{ID}}$ of the Routing Table Set

We find the worst-case reduction ratio for defining the total order $\leq_{\mathrm{ID}}$ in FRT-2-Chord. Let E be the routing table of a node s. A routing table E satisfies $i < j \Rightarrow \mathrm{d}_{\mathrm{chord}}(s, e_i) < \mathrm{d}_{\mathrm{chord}}(s, e_j), (e_i, e_j \in E)$. Here, we will focus on the worst-case reduction ratio of a forward to a node e_i. Let m_i be the median value of the path from e_i to e_{i+1} in the clockwise direction. FRT-2-Chord routes a query from e_i to t in the clockwise direction or in the anticlockwise direction. If clockwise, the reduction ratio of a forward to e_i takes the worst-case value when $t = m_i$. If anticlockwise, the worst-case value occurs when $t = m_{i-1}$. Let $r_i^{\mathrm{cw}}(E)$ be the worst-case reduction ratio of forwarding to a node e_i in the clockwise direction; let $r_i^{\mathrm{acw}}(E)$ be the worst-case reduction ratio of forwarding to a node e_i in the anticlockwise direction. Fig. 1 shows the relation among positions e_{i-1}, e_i, and e_{i+1} in the clockwise or anticlockwise directions from s.

We define a set of the worst-case reduction ratio $\{r_i(E)\}$ for defining a total order as follows.

Definition 5:

$$\{r_i(E)\} = \{r_i^{\mathrm{cw}}(E)\} \cup \{r_i^{\mathrm{acw}}(E)\} \tag{5}$$

Let $\{r_{(i)}(E)\}$ be the list of reduction ratios arranged in descending order. We define a total order $\leq_{\mathrm{ID}}$ in FRT-2-Chord as follows.

Definition 6:

$$E \leq_{\mathrm{ID}} F \Leftrightarrow \{r_{(i)}(E)\} \leq_{\mathrm{dic}} \{r_{(i)}(F)\} \tag{6}$$

In this definition, $\leq_{\mathrm{dic}}$ is the lexicographical order operator.

Let e_k be the entry that is on the path from s to $s+2^{m-1}$ in the clockwise direction and is the closest to $s+2^{m-1}$. Now, we describe $r_i^{\mathrm{cw}}(E)$ and $r_i^{\mathrm{acw}}(E)$ in detail. The following theorem holds.

Theorem 1:

$$\{r_i^{\mathrm{cw}}(E)\} \cup \{r_i^{\mathrm{acw}}(E)\} = \{r_i^{\mathrm{cw}}(E)\}. \tag{7}$$

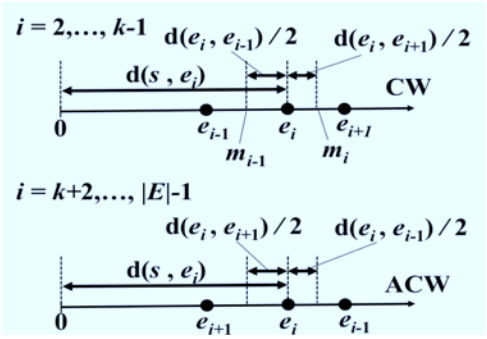

Fig. 1. Relation among positions of entries in clockwise and anticlockwise directions from s.

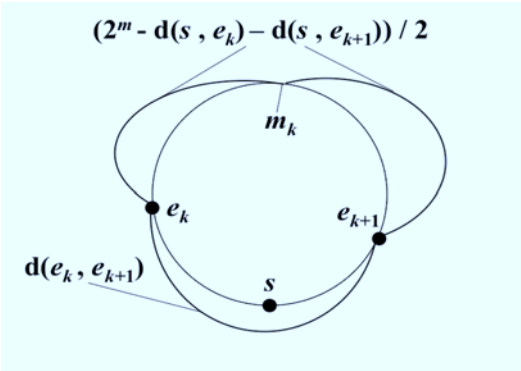

Fig. 2. Example of relation among positions of s, e_k, and e_{k+1} where $\mathrm{d}(e_k, m_k) \neq \mathrm{d}(e_k, e_{k+1})/2$ holds.

Proof: The calculation of $\mathrm{d}(e_i, m_i)$ when $i \neq \mathrm{k}$ is different from the one when $i = k$. When $i \neq k$, $\mathrm{d}(e_i, m_i) = \mathrm{d}(e_i, e_{i+1})/2$. Otherwise, $\mathrm{d}(e_i, m_i) = 2^m - \mathrm{d}(s, e_i) - \mathrm{d}(s, e_{i+1})$. Fig. 2 shows an example of the relation between positions; in this example case, $\mathrm{d}(e_k, m_k) \neq \mathrm{d}(e_k, e_{k+1})/2$ holds.

$r_i^{\mathrm{cw}}(E)$ and $r_i^{\mathrm{acw}}(E)$ are defined as follows.

$$r_i^{\mathrm{cw}}(E) = \begin{cases} \dfrac{|\mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_i)|}{\mathrm{d}(s, e_{i+1}) + \mathrm{d}(s, e_i)} & (i \neq k) \\ \dfrac{2^m - \mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_i)}{2^m - |\mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_i)|} & (i = k) \end{cases} \tag{8}$$

$$r_i^{\mathrm{acw}}(E) = \begin{cases} \dfrac{|\mathrm{d}(s, e_i) - \mathrm{d}(s, e_{i-1})|}{\mathrm{d}(s, e_i) + \mathrm{d}(s, e_{i-1})} & (i \neq k+1) \\ \dfrac{2^m - \mathrm{d}(s, e_i) - \mathrm{d}(s, e_{i-1})}{2^m - |\mathrm{d}(s, e_i) - \mathrm{d}(s, e_{i-1})|} & (i = k+1) \end{cases} \tag{9}$$

Clearly,

$$r_i^{\mathrm{cw}}(E) = r_{i+1}^{\mathrm{acw}}(E). \tag{10}$$

Therefore, (7) holds. ■

Then, we let $r_i(E)$ be $r_i^{\mathrm{cw}}(E)$.

C. Guarantee of Reachability

FRT-2-Chord and FRT-Chord have different handling of sticky entries and guarantees of reachability. Because of the two directions property, in addition to the successor list and predecessor, sticky entries include a *predecessor list*, which is a list containing a certain number of the nodes closest in the anticlockwise direction.

D. Entry Filtering

In FRT-2-Chord, we define the *updated worst-case reduction ratio* R_i^E when e_i is removed as follows.

Definition 7:

$$R_i^E = \begin{cases} \dfrac{|\mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_{i-1})|}{\mathrm{d}(s, e_{i+1}) + \mathrm{d}(s, e_{i-1})} & (i \neq k, i \neq k+1) \\ \dfrac{2^m - \mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_{i-1})}{2^m - |\mathrm{d}(s, e_{i+1}) - \mathrm{d}(s, e_{i-1})|} & (i = k, k+1) \end{cases} \tag{11}$$

Entry filtering in FRT-2-Chord is summarized as follows.

1) Substitute entries in E into C
2) Remove sticky entries from C
3) Select the entry e_i from C that minimizes R_i^E.

By sorting R_i^E into ascending order, the entry to be removed can be found efficiently. The following theorems of FRT-Chord hold for FRT-2-Chord as well.

Theorem 2: In FRT-2-Chord, let $E - \{e_{i^*}\}$ be a routing table filtered by removing e_{i^*}. For any e_i which is not a sticky entry,

$$E - \{e_{i^*}\} \leq_{\mathrm{ID}} E - \{e_i\}. \tag{12}$$

Proof: We are focusing on the updated worst-case reduction ratio. Let l^* and l be as follows.

$$l^* = \min\{j | R_{i^*}^E > r_{(j)}(E)\} \tag{13}$$

$$l = \min\{j | R_i^E > r_{(j)}(E)\} \tag{14}$$

A list of $\{r_{(i)}(E - \{e_{i^*}\})\}$ from the beginning to $l^* + 1$ is given by $r_{(1)}(E), r_{(2)}(E), ..., r_{(l^*-1)}(E), R_{i^*}^E, r_{(l^*)}(E)$. Then, the list of $\{r_{(i)}(E - \{e_{i^*}\})\}$ from the beginning to $l^* - 1$ is preserved across entry filtering. Similarly, a list of $\{r_{(i)}(E - \{e_i\})\}$ from the beginning to $l + 1$ is given by $r_{(1)}(E), r_{(2)}(E), ..., r_{(l-1)}(E), R_i^E, r_{(l)}(E)$. For any j,

$$\{r_i(E - \{e_j\})\} = \{r_i(E)\} \cup \{R_j^E\} - \{r_j(E), r_{j-1}(E)\} \tag{15}$$

holds. Futhermore,

$$R_j^E > r_j(E), R_j^E > r_{j-1}(E) \tag{16}$$

hold. Therefore, a list of $\{r_{(i)}(E - \{e_i\})\}$ from the beginning to $l - 1$ is preserved across entry filtering. Since $R_{i^*}^E \leq R_i^E$ holds by definition of i^*, $l^* \geq l$ also holds. Therefore,

$$r_{(i)}(E - \{e_{i^*}\}) = r_{(i)}(E - \{e_i\}) (i = 1, 2, ..., l - 1) \tag{17}$$

and

$$r_{(l)}(E - \{e_{i^*}\}) \leq r_{(l)}(E - \{e_i\}) \tag{18}$$

hold. Hence, (12) holds. ■

It is possible that routing table refinement through repeated entry learning and entry filtering will stop when any most recently learned entry is selected as the entry to remove. Such a routing table E is called a *convergent routing table*, and the following theorem holds.

Theorem 3: In FRT-2-Chord, assuming that all nodes have convergent routing tables with $\mathrm{O}(\log N)$ entries in an N-nodes network, path lengths are $\mathrm{O}(\log N)$ with high probability.

Proof: S_i^E is defined as follows.

$$S_i^E = \begin{cases} \log \dfrac{\mathrm{d}(s, e_{i+1})}{\mathrm{d}(s, e_i)} & (i = 1, 2, ..., k-1) \\ \log \dfrac{\mathrm{d}(s, e_i)}{\mathrm{d}(s, e_{i+1})} & (i = k+1, ..., |E|-1) \end{cases} \tag{19}$$

$$S_k^E = \begin{cases} \log \dfrac{2^m - \mathrm{d}(s, e_k)}{\mathrm{d}(s, e_{k+1})} & (\mathrm{d}(s, e_{k+1}) \leq \mathrm{d}(s, e_k)) \\ \log \dfrac{2^m - \mathrm{d}(s, e_{k+1})}{\mathrm{d}(s, e_k)} & (\mathrm{d}(s, e_{k+1}) > \mathrm{d}(s, e_k)) \end{cases} \tag{20}$$

Let J be a set of indices j, where a node exists in the range from e_j to e_{j+1} in the clockwise direction; let L be a set of indices l, where a node exists in the range from e_{l-1} to e_l in the clockwise direction; let P be a set of indices p otherwise. By definition,

$$r_i(E) = 1 - 2/(2^{S_i^E} + 1) \tag{21}$$

holds for any $i, j = 1, ..., |E|-1$. Therefore, if $r_i(E) < r_j(E)$ holds, then $S_i^E < S_j^E$ holds for i.

$$R_i^E = 1 - 2/(2^{S_i^E + S_{i-1}^E} + 1) \tag{22}$$

holds for any $i, j = 2, 3, ..., k-1, k+2, ..., |E|-1$. Therefore, if $R_i^E < R_j^E$ holds, then $S_i^E + S_{i-1}^E < S_j^E + S_{j-1}^E$ holds for any $i, j = 2, 3, ..., k-1, k+2, ..., |E|-1$. Since all nodes have convergent routing tables,

$$r_j^{\mathrm{cw}}(E) = r_j(E) \leq R_i^E \tag{23}$$

for any $j \in J$ and for any $i = 2, ..., |E| - 1$. Similarly,

$$r_l^{\mathrm{acw}}(E) = r_{l-1}(E) \leq R_i^E \tag{24}$$

holds for any $l \in L$ and for any $i = 2, ..., |E| - 1$. With high probability, the distance between two generic consecutive nodes is at least $2^m/N^2$ [15]. For any $j \in J$, the following inequality holds.

$$S_j^E \leq \frac{\sum_{2 \leq i \leq |E|-1, i \neq k, i \neq k+1} (S_i^E + S_{i-1}^E)}{|E| - 4} \tag{25}$$

$$< \frac{8}{|E| - 4} \log N \tag{26}$$

Thus, for $|E| = 4 + (8/\log 3) \log N$,

$$r_j(E) < 1 - \frac{2}{N^{8/|E|-4} + 1} = \frac{1}{2} \tag{27}$$

Similarly, $r_{l-1}(E) < 1/2$ holds for $|E| = 4 + (8/\log 3) \log N$. We now consider the upper limit of path lengths needed to reduce the remaining distance to $2^m/N$ or less. If the message is forwarded to $e_p (p \in P)$, the responsible node is e_p, and the routing will terminate. We focus on the case where each node forwards to $e_i (i \in J$ or $i \in L)$. Because the remaining distance is obviously 2^{m-1} or less, and, the worst-case reduction ratio is under $1/2$ when the message is forwarded to a node in either the clockwise direction or anticlockwise direction, the path length is $\log N - 1$. When the remaining distance is at most $2^m/N$, the number of node identifiers landing in a range of this size is $\mathrm{O}(\log N)$ with high probability [8]. Thus, the query reaches the destination node within an additional $\mathrm{O}(\log N)$ hops. Therefore, the entire path length is $\mathrm{O}(\log N)$. ■

E. GFRT-2-Chord

GFRT-2-Chord is an extension of FRT-2-Chord in order to consider node groups. Its methods of guarantee of reachability and entry learning are the same as FRT-Chord. It is characterized by its entry filtering. We define the following variables for the routing table $E = \{e_i\}$ at a node s.

- $E_G = \{e \in E \mid e.\mathrm{group} = s.\mathrm{group}\}$
- $E_{\bar{G}} = \{e \in E \mid e.\mathrm{group} \neq s.\mathrm{group}\}$
- e_α is the closest entry in E_G from s in the clockwise direction.
- e_β is the closest entry in E_G from s in the anticlockwise direction.
- $E_{\mathrm{near}} = \{e \in E \mid \mathrm{d}(s, e) < \mathrm{d}(s, e_\alpha) \vee \mathrm{d}(s, e) < \mathrm{d}(s, e_\beta)\}$
- $E_{\mathrm{far}} = \{e \in E \mid \mathrm{d}(s, e_\alpha) \leq \mathrm{d}(s, e) \wedge \mathrm{d}(s, e_\beta) \leq \mathrm{d}(s, e)\}$
- $E_{\mathrm{leap}} = E_{\mathrm{far}} \cap E_{\bar{G}}$

In GFRT-2-Chord, sticky entries are a successor list, a predecessor list, a *group successor list* and a *group predecessor list*. The group successor list and the group predecessor list at a node s means a successor list and a predecessor list respectively in a network limited to nodes belonging to the same group as s.

GFRT-2-Chord performs entry filtering as follows:

1) Substitute entries in $E_{\bar{G}}$ into C if $E_{\mathrm{leap}} \neq \emptyset$, otherwise substitute entries in E into C.
2) Remove sticky entries from C
3) Select the entry e_i from C that minimizes R_i^E.

IV. Evaluation

We implemented FRT-2-Chord on Overlay Weaver [16] [17], an overlay construction toolkit, and performed experiments. In all experiments the routing table size is 160, the size of the successor list is 4 (FRT-2-Chord and FRT-Chord), and the size of the predecessor list is 4 (FRT-2-Chord). We employ queries for iterative lookup in all experiments.

A. Path Lengths in FRT-2-Chord

After N nodes join a system, each node sends a query 200 times; each query is sent to a randomly chosen key by a randomly chosen node ($N = 10^2, 10^3, 10^4$). Fig. 3 shows the average path length for N queries; these data reveal that average path lengths and number of lookups are

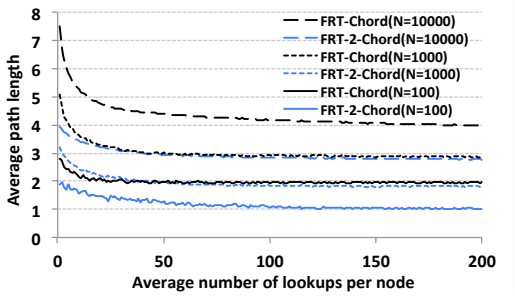

Fig. 3. Change of average path length along with the number of lookups per node.

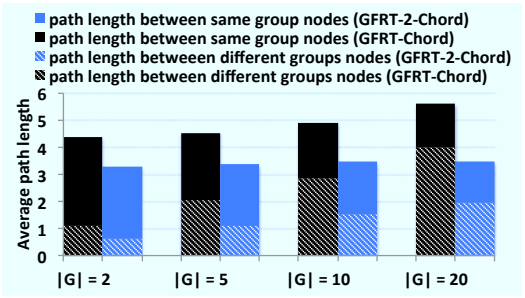

Fig. 4. Average path length.

both reduced in FRT-2-Chord and FRT-Chord. FRT-2-Chord repeatedly refines routing tables by entry learning and entry filtering, and average path length in FRT-2-Chord is shorter than in FRT-Chord.

When N is 10^2 the routing table size is larger than the number of nodes. Thus, each node is able to have entries for of all the nodes. In this experiment, after 200 lookups from each node, average path length is 1.01 in FRT-2-Chord. After additional 1000 lookups from each node, average path length approaches 1.00.

B. Path Lengths in GFRT-2-Chord

We also implemented GFRT-2-Chord on Overlay Weaver. After N nodes each of which belong to a group chosen randomly join a system, each node sends a query 200 times. In the experiments, we set $|E| = 20, N = 1000$ and size of the group successor list and size of the group predecessor list are 1. Fig. 4 shows average path length average path length between different group nodes.

The forwarding mechanism in GFRT-2-Chord is able to be regarded as removing the last hop to the node responsible for a key in GFRT-Chord. In experiments, path length in GFRT-2-Chord is more than one hop shorter than in GFRT-Chord. Therefore, the reduction of path length in GFRT-2-Chord derives from not only the forwarding mechanism but also the routing table construction with the identifier distance.

V. Conclusion

In this paper, we proposed FRT-2-Chord, an FRT-based DHT. FRT-2-Chord can adapt factors such as number of nodes, network proximity, and node groups. Because of this, FRT-2-Chord efficiently routes in a wider range of situations than existing DHTs do.

References

[1] BitTorrent, http://www.bittorrent.com/.

[2] B. Cohen, "Incentives build robustness in bittorrent," in *1st Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[3] K. Jünemann, "Dsn research group-live monitoring," 2011, http://dsn.tm.uni-karlsruhe.de/english/2936.php.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *Proc. SOSP '07*, vol. 41, no. 6. ACM, 2007, pp. 205–220.

[5] Apache Cassandra, The Apache Software Foundation, http://cassandra.apache.org/.

[6] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[7] Apache Hadoop, The Apache Software Foundation, http://hadoop.apache.org/.

[8] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Proc. IFIP/ACM Middleware 2001*, pp. 329–350, 2001.

[10] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Proc. IPTPS '02*, pp. 53–65, 2002.

[11] V. Mesaros, B. Carton, and P. Van Roy, "S-chord: Using symmetry to improve lookup efficiency in chord," in *Proc. PDPTA '03*, 2003.

[12] B. Leong, B. Liskov, and E. Demaine, "Epichord: parallelizing the chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006.

[13] P. Fonseca, R. Rodrigues, A. Gupta, and B. Liskov, "Full-information lookups for peer-to-peer overlays," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 20, pp. 1339–1351, 2009.

[14] H. Nagao and K. Shudo, "Flexible routing tables: Designing routing algorithms for overlays based on a total order on a routing table set," in *Proc. IEEE P2P '11*, 2011, pp. 72–81.

[15] G. Cordasco and A. Sala, "2-chord halved," *Proc. HOT-P2P '05*, pp. 72–79, 2005.

[16] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay weaver: An overlay construction toolkit," *Computer Communications*, vol. 31, no. 2, pp. 402–412, 2008.

[17] K. Shudo, Overlay Weaver: An Overlay Construction Toolkit, http://overlayweaver.sourceforge.net/.