FRT-Skip Graph: A Skip Graph-Style Structured Overlay based on Flexible Routing Tables

Masashi Hojo[†*] , Ryohei Banno[‡†] and Kazuyuki Shudo[†]

[†]Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152-8552, Japan
[‡]NTT Network Innovation Laboratories, NTT Corporation
Midori-cho 3-9-11, Musashino-shi, Tokyo 180-0012, Japan
Email: hojo.m.aa@m.titech.ac.jp, banno.ryohei@lab.ntt.co.jp, shudo@is.titech.ac.jp

***Abstract*—Structured overlays enable a number of nodes to construct a logical network autonomously and search each other. Skip Graph, one of the structured overlays, constructs an overlay network based on Skip List structure and supports range queries for keys. Skip Graph manages routing tables based on random digits; therefore, the deviation of them disturbs effective utilization of the routing table entries and increases path length than the ideal value. We therefore propose FRT-Skip Graph, a novel structured overlay that solves the issues of Skip Graph and provides desirable features not in Skip Graph. FRT-Skip Graph is designed based on Flexible Routing Tables and supports range queries similarly to Skip Graph. Furthermore, it provides features derived from FRT, namely, dynamic routing table size and high extensibility.**

I. Introduction

A structured overlay constructs an application layer network composed of nodes on a computer network such as Internet. Each node in an overlay has its identifier (ID) and the nodes construct a logical network based on their IDs. Enabling nodes to search each other is achieved by relaying messages for target IDs. Each node determines a next hop to which the node relays a message by referring its routing table. A routing table holds pairs of a node ID and an IP address. In consequence of such a routing mechanism, structured overlays prevent inefficient communications like flooding and they are highly scalable.

Distributed hash table (DHT) [1], [2], [3] is an application of structured overlays. DHT provides two functions that put and get data in (key, value) format. Each data has a data ID determined by a hash value for its key, and the data is stored in a node that manages the range of IDs including the data ID. A range of IDs of a node is determined by its ID and neighbors' IDs. Put/get functions are performed by relaying queries to a responsible node of the data ID on a overlay. Since IDs do not preserve the order of keys due to the hash function, flexible searches such as range queries are not supported in DHT.

In contrast to DHT, Skip Graph [4] is a structured overlay that constructs a network while keeping the key order. Skip Graph has a network topology like Skip List [5], a data structure of a extended linked list, and performs searching data in $O(\log N)$ of path length similarly to many of DHT algorithms. Furthermore, since it keeps the key order, it supports range queries for keys. In Skip Graph, each node constructs shortcut links based on random digits for keeping path length short while retaining the order of keys. However, introducing random digits for shortcut links often increases path length.

Based on these advantages and disadvantages of Skip Graph, we propose FRT-Skip Graph, a novel structured overlay that constructs a network similar to Skip Graph and keeps path length shorter than it. While it uses random digits to construct a routing table, FRT-Skip Graph resolves the problem of Skip Graph; usuless entries of a routing table. Furthermore, since FRT-Skip Graph is designed based on Flexible Routing Tables [6], it provides features derived from FRT, dynamic routing table size and high extensibility.

In this paper, We describe the design of FRT-Skip Graph, and prove that its routing is performed in $O(\log N)$ of path length. Moreover, we observe its flexibility from FRT and further improvement of its construction of a routing table.

II. Related Work

In this section, first, we describe Skip Graph, a structured overlay that is a base algorithm of our proposal overlay. Next, we explain FRT, a scheme for designing routing algorithms for overlay networks.

A. Skip Graph

Skip Graph [4] is a structured overlay that has a topology like Skip List [5], one of the data structures. Skip List is a randomized algorithm that constructs shortcut links stochastically on a linked list for the sake of efficient search. Skip Graph constructs such a topology by a large number of nodes in an autonomous distributed way, and as well as Skip List, it manages keys with a total ordering while holding the order.

* Present affiliation: NTT DATA Corporation, Toyosu Center Building, Toyosu 3-3-3, Koto-ku, Tokyo 135-6033, Japan.

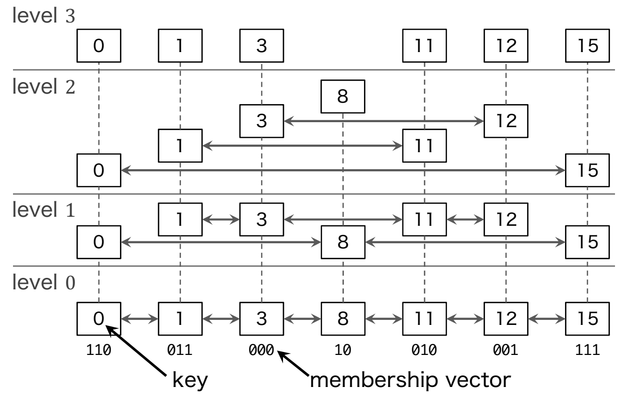

Fig. 1: An example of Skip Graph.

Therefore, Skip Graph performs range queries for keys that cannot be supported in DHT.

Figure 1 shows an example of a network structure of Skip Graph. In Skip Graph, each node maintains one key. Nodes are arranged in the order of their keys, and all nodes compose a network with bidirectional links at level 0. Additionally, nodes construct shortcut links at each level higher than level 0. To determine a pair of nodes connected with eath other, sequence of m-ary random digits, namely, membership vector (MV) is used (in this paper, let m be 2). A MV of a node is independent of its key. At level i, shortcut links are constructed by nodes with i common prefixes of their MVs. According to this method, the number of links, in other words, routing table size of each node is $O(\log N)$, where N is the number of nodes in the whole of overlay.

Routing in Skip Graph is similar to Skip List. Routing starts at the highest level and approaches a target key at the level. If the message cannot approach any more at the level, dropping to the next level, the routing continues. This way of routing can skip many nodes at a high level to approach the target quickly, and it keeps path length within $O(\log N)$.

In Skip Graph, since nodes construct their routing tables by MVs, inefficient routing tables often appear. For example, a situation that two nodes maintaining close keys have MVs with long common prefixes causes to construct shortcut links over high- and low-levels between only the two nodes. In other words, entries in a routing table prepared for each level are filled with one node. In such a situation, many nodes cannot be skiped at a high level in routing, and path length becomes longer.

B. Flexible Routing Tables (FRT)

Flexible Routing Tables (FRT) is a scheme for designing routing algorithms for overlay networks. Whereas many conventional structured overlays strictly define a routing table should be built as a combination of node IDs and construct it by finding closest nodes to the IDs, FRT proposes an entirely different routing table constructing scheme. That is, an overlay based on FRT defines a total order of the routing table set $\leq_{\mathrm{RT}}$, and improves a routing table in accordance with it. In contrast to conventional structured overlays that restrict candidates for a routing table, FRT-based overlays provide various advantages explained in Sec. II-B1.

1) Advantageous Features of FRT: The algorithms designed based on FRT have the following advantages:

- It is possible to adjust routing table size L dynamically. L is adjusted in response to node's capacity or network stability.
- It is possible to design an algorithm that consistently handles both single-hop routing and multi-hop routing. If a node can hold all other nodes in its routing table ($N \leq L$, where N is the number of nodes), the node can forward a message in single-hop. Otherwise ($N > L$), a message is relayed node by node.
- FRT-based algorithms are extensible to consider various indices of routing performance. For example, Proximity-aware Flexible Routing Tables (PFRT) [7] considers network proximity, and Grouped Flexible Routing Tables (GFRT) [8] considers node groups. Flow-based Flexible Routing Tables (FFRT) [9] constructs a routing table by measuring flow rate of messages.

2) Routing Table Management Operations: To construct and to improve a routing table, FRT prepares two operations as follows.

- *Entry learning*: Entry learning is an operation to insert node's information into a routing table. Executing this operation, a node inserts a new other node's information into its routing table without screening. The node's information is available from various communication results, and in addition, it is possible to get it by active lookup.
- *Entry filtering*: Entry filtering is an operation to remove an entry from a routing table. When the number of entries in a routing table exceeds L, this operation is executed in order to retain a limitation of routing table size. The entry is selected according to total order of the routing table set described in Sec. II-B3.

In FRT-based routing table construction, at first, entry learning is repeated until exceeding L. Then, a couple of entry learning and entry filtering is executed continuously and the routing table is refined gradually.

3) $\leq_{\mathrm{RT}}$: Total Order of the Routing Table Set: A FRT-based overlay defines a total order of the routing table set $\leq_{\mathrm{RT}}$ to determine a better one of two routing tables. Between two routing tables E and F, if we have $E \leq_{\mathrm{RT}} F$, then we determine E is better than F. A FRT-based structured overlay improves routing efficiency by refining routing tables accordingly $\leq_{\mathrm{RT}}$. We can design a FRT-based overlay by defining $\leq_{\mathrm{RT}}$ considering a topology and a distance function of the overlay.

4) Structured Overlays based on FRT: Several structured overlays based on FRT are proposed to this day. FRT-Chord [6], FRT-2-Chord [10] and FRT-Chord$^{\#}$ [11] have one-dimensional circular ID spaces as their topologies. A two-dimensional structured overlay based on FRT is also proposed [12].

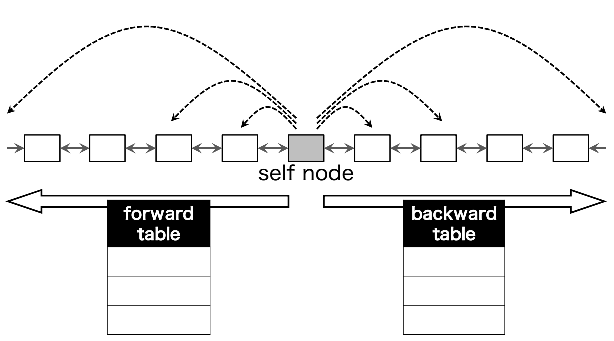

Fig. 2: A routing table of FRT-Skip Graph.

III. FRT-Skip Graph

In this section, we propose FRT-Skip Graph, a novel structured overlay that constructs a Skip Graph-style network to enable range queries and solves the problem in Skip Graph to keep path length short. In FRT-Skip Graph, each node has a key and a MV similarly to Skip Graph, and its topology is also based on Skip List structure. On the other hand, differently from Skip Graph, shortcut links are not necessarily bidirectional links. That is, when node A constructs a shortcut link to node B, B does not need to construct a link to A. Instead, each node refining its routing table, these links eventually become symmetrical.

Such a routing table construction is based on Flexible Routing Tables (FRT). Since we design FRT-Skip Graph based on FRT, it provides high extensibility to improve routing performance while keeping path length short. For example, we can design an extended algorithm to execute range queries and broadcast efficiently by holding nodes maintaining close keys in a routing table. Besides, holding nodes with low network latency preferentially, we aim to improve routing performance from the point of view of network proximity.

A. Routing Table Construction

In this section, first, we explain structure of a routing table of FRT-Skip Graph. Next, we define sticky entry and total order of the routing table set in FRT-Skip Graph to construct a routing table based on FRT.

1) Structure of Routing Table: Each node in FRT-Skip Graph is arranged in the order of keys and manages links both in the direction of smaller keys (to forward nodes) and in the direction of larger keys (to backward nodes). Therefore, a node manages two independent routing tables to insert both forward nodes and backward nodes into. Hereinafter, they are called forward table (FT) and backward table (BT).

Figure 2 shows structure of a routing table in FRT-Skip Graph. Both FT and BT have L entries, where L is specified independently by the node. Each entry contains node information; a key, an IP address and a MV. In contrast to Skip Graph in which routing table entries are assigned to each level and a same node can be inserted into two or more entries, FRT-Skip Graph constructs a flat routing table; each entry has a different node from each other. Therefore, one node never occupys more than one entry. Such a design of a routing table solves the problem of Skip Graph; a shortcut link at level i (the ith entry of a routing table) has a close node and path length becomes longer than the ideal case.

2) Sticky Entry: FRT-based structured overlays define some entries as *sticky entries*. Sticky entries are excluded from removal candidates for entry filtering. In FRT-Skip Graph, we define two entries with the closest key from its own in both FT and BT as sticky entries . All nodes hold their sticky entries and form a doubly linked list at level 0, so that reachability to any nodes is guaranteed. To enhance the churn tolerance, we can assign several closest nodes for sticky entries and design to construct bidirectional links between several close nodes.

3) $\leq_{\text{FRT-SG}}$: Total Order of the Routing Table Set: In FRT-Skip Graph, the superiority of a routing table is determined by closeness to the self node and the level of each entry. Hereinafter, we express both FT and BT as $E = \{e_i\}_{i=1,\dots,|E|}$ and a level of an entry e as $e.\text{lv}$. In E, each node is assigned an index in increasing order of closeness to the self node. In this paper, we use closeness to the self node for ordering relation between two other nodes.

To define the total order of the routing table set of FRT-Skip Graph, first, we observe a routing table of Skip Graph. Levels defined in Skip Graph are used to estimate the number of nodes between a self node and the other nodes in a linked list where all nodes participate (in other words, a list at level 0) with a stochastic approach. To be specific, an entry at level i is expected to be a distant node at an interval of 2^i from the self node. In Skip Graph, a node maintains one entry at each level for both two directions and constructs shortcut links to distant nodes that are expected to be at intervals of $2^0, 2^1, \dots, 2^i$.

In FRT-Skip Graph, we can expect equivalent efficiency of path length to Skip Graph by designing a routing table to contain each entry that is at each level of Skip Graph. Furthermore, by focusing the problem of Skip Graph—an entry at level i often cannot skip 2^i nodes—, we presume that it is desirable to contain several high level nodes preferentially to increase efficiency of shortcut links.

From the above observations, we consider an ideal routing table of FRT-Skip Graph as a routing table that holds the same order of the number of entries at each level and depending on routing table size, holds more entries at high levels preferentially. To construct such a routing table, preparing some expressions, we define the routing table set of FRT-Skip Graph $\leq_{\text{FRT-SG}}$.

First, we define E_l, a part of E determined by entries' level.

Definition 3.1 (a partition of a routing table E according to levels):

$$E_l = \{e \in E | e.\text{lv} = l\}. \tag{1}$$

Next, we define a sequence by focusing the number of members of E_l, in other words, the number of entries with level l in E.

Definition 3.2 ($\mathrm{seq}(E)$*: a sequence of levels in a routing table E):*

$$\mathrm{seq}(E) = (-l_1, -l_2, \dots), \qquad (2)$$
$$\forall i < \forall j, \quad (|E_{l_i}| > |E_{l_j}|) \lor (|E_{l_i}| = |E_{l_j}| \land l_i > l_j).$$

For example, when $E = E_0 \coprod E_1 \coprod E_2 \coprod E_3$, $|E_0| = 3$, $|E_1| = 4$, $|E_2| = 4$ and $|E_3| = 1$, we have $\mathrm{seq}(E) = (-2, -1, 0, -3)$.

Definition 3.3 ($(E)_{\mathrm{lv}}$*: a sequence of entries in a routing table E according to levels):* $(E)_{\mathrm{lv}}$ is a sequence of entries in a routing table E, in which each entry is arranged in decreasing order of indices of members of E_{l_i}, and each E_{l_i} is arranged in increasing order of i of E_{l_i}.

For example, when $E = E_0 \coprod E_1 \coprod E_2 \coprod E_3$, $|E_0| = 3$, $|E_1| = 4$, $|E_2| = 4$ and $|E_3| = 1$ (the similar situation to one mentioned above), we have $(E)_{\mathrm{lv}}$ is arranged in E_2, E_1, E_0, E_3, and entries in each E_l are arranged in decreasing order of their indices.

Then, using these definitions, the total order of the routing table set is defined as follows.

Definition 3.4 ($\leq_{\text{FRT-SG}}$*: the total order of the routing table set):*

$$\begin{aligned} E \leq_{\text{FRT-SG}} F \quad \Leftrightarrow \quad & (\mathrm{seq}(E) <_{\mathrm{dic}} \mathrm{seq}(F)) \\ & \lor\{(\mathrm{seq}(E) =_{\mathrm{dic}} \mathrm{seq}(F)) \\ & \qquad \land((E)_{\mathrm{lv}} \leq_{\mathrm{dic}} (F)_{\mathrm{lv}})\}. \end{aligned} \qquad (3)$$

Where $\leq_{\mathrm{dic}}$ is lexicographical order:

$$\begin{aligned} \{a_i\} <_{\mathrm{dic}} \{b_i\} \quad &\Leftrightarrow \quad a_k < b_k \quad (k = \min\{i \mid a_i \neq b_i\}), \\ \{a_i\} =_{\mathrm{dic}} \{b_i\} \quad &\Leftrightarrow \quad a_i = b_i, \\ \{a_i\} \leq_{\mathrm{dic}} \{b_i\} \quad &\Leftrightarrow \quad (\{a_i\} <_{\mathrm{dic}} \{b_i\}) \lor (\{a_i\} =_{\mathrm{dic}} \{b_i\}). \end{aligned}$$

In entry filtering of FRT-Skip Graph, an entry $e^* \in E^*$ satisfying the following condition is deleted by using $\leq_{\text{FRT-SG}}$ described above. Where E^* is a set of entries in E without sticky entries.

$$E \setminus \{e^*\} \leq_{\text{FRT-SG}} E \setminus \{e\}, \quad e \in E^*. \qquad (4)$$

B. Routing

Routing in Skip Graph is performed by forwarding to an entry that approaches closest to a target key as dropping levels. On the other hand, routing in FRT-Skip Graph is performed by only selecting an entry from all members in the routing table that approaches closest to the target key; that is, greedy routing. In contrast to Skip Graph that selects a forwarding entry based on levels, FRT-Skip Graph selects the best entry from all entries, and together with utilization all entries without waste, FRT-Skip Graph keeps path length shorter than Skip Graph.

To reveal that path length of FRT-Skip Graph achieves to become equivalent to or shorter than Skip Graph, first, we prepare several lemmas.

Lemma 3.1: In a network of FRT-Skip Graph constructed with the number of nodes N, the number of entries in a routing table derived from Skip Graph is at most $O(\log N)$.

Proof : In Skip Graph, the number of entries in a routing table is length of the longest match of MV, in other words, the value of highest level. As shown in Sec. II-A, the value is $O(\log N)$, and moreover, a same node may practically be inserted two or more entries. Therefore, the number of different nodes is not more than the number of entries in Skip Graph, that is,$O(\log N)$. □

Lemma 3.2: In a network of FRT-Skip Graph constructed with the number of nodes N, letting routing table size be $L^* = O(\log N)$, entries derived from Skip Graph are not deleted by entry filtering of FRT-Skip Graph.

Proof : Let routing table size be $L^* = O(\log N)$. From lem. 3.1, it is enough to contain all entries derived from Skip Graph. Therefore, here we prove that these entries are not deleted by entry filtering according to $\leq_{\mathrm{FRT-SG}}$. From Def. 3.4, when deleted according to $\leq_{\mathrm{FRT-SG}}$, an entry is selected which is the most distant node at the level that has the most entries in all levels. Since entries derived from Skip Graph are closest node at their level, it is not selected to delete. □

Using these lemmas, we have the following theorem.

Theorem 3.1: In a network of FRT-Skip Graph constructed with the number of nodes N, letting routing table size be $L^* = O(\log N)$, path length becomes $O(\log N)$ after improvement of a routing table by a sufficient number of times.

Proof : From Lem. 3.2, entries derived from Skip Graph that are inserted into a routing table once are not deleted by entry filtering. Therefore, here we discuss a situation that all of these entries are inserted into a routing table after improvement of a routing table by a sufficient number of times. In such a situation, greedy routing adopted in FRT-Skip Graph selects an entry to forward that are either an entry selected in routing of Skip Graph or an entry approach to a target closer than it. Since the routing in this policy repeated by each node keeps path length equivalent to or shorter than Skip Graph, path length in FRT-Skip Graph is $O(\log N)$. □

Theorem 3.1 shows that path length becomes $O(\log N)$ with letting routing table size in each node be $O(\log N)$. In FRT-Skip Graph, the routing table size can be specified to larger value depending on stability of the network or node's capacity, and it is possible to configure to perform single-hop routing as referred in Sec. II-B1.

C. Skip Graph Emulation

A routing table of FRT-Skip Graph can include all entries derived from Skip Graph with enough the size of it. Therefore, FRT-Skip Graph can emulate behavior of Skip Graph by switching routing algorithm from greedy routing of FRT-Skip Graph to one of Skip Graph. Emulation of Skip Graph allows FRT-Skip Graph to be compatible with Skip Graph and to apply to proposed applications of Skip Graph.

IV. Evaluation

We implemented the proposed overlay on Overlay Weaver [13], [14], an overlay construction toolkit, and performed experiments.

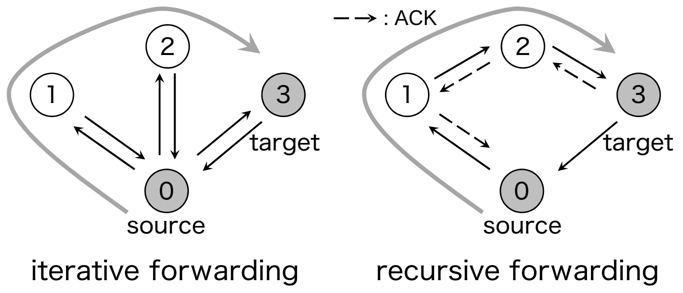

Fig. 3: Two styles to forward queries.

In experiments shown below, we adopted iterative style to forward queries that is performed by repeating to query a present node about next nodes by the initiating node. As shown in Fig. 3, there are two styles to forward queries, namely, iterative forwarding and recursive forwarding. In iterative forwarding, the initiating node can notice not only the target node but also the nodes included in the route. Moreover, after construction a routing table by lookups of the number of 200 times by each node, we measured path length of the lookups for uniform random nodes.

A. *Comparison with Skip Graph*

We brought emulated Skip Graph for comparison referred in Sec. III-C to evaluate routing performance of FRT-Skip Graph. Emulated Skip Graph constructs a routing table similarly to FRT-Skip Graph; however, it uses only entries derived from Skip Graph in routing. We also note specifically in these experiments that it does not adopt level-based routing of Skip Graph but greedy routing similarly to FRT-Skip Graph. Path length of this style of routing is equivalent to or shorter than original Skip Graph, but not longer. In this experiment, each node inserts all entries derived from Skip Graph by improvement of a routing table by a sufficient number of times to arrange the condition similar to Skip Graph. In the following, Emulated Skip Graph is simply called Skip Graph.

In this experiment, we specified the number of nodes to $N = 100, 1000, 10000$ and routing table size of FRT-Skip Graph to $L = 7, 10, 14$, respectively—they are determined as $\log N$ brought from routing table size of Skip Graph.

Figure 4 and Fig. 5 show the results of this experiment. The averages and 99 percentiles of path length were calculated from 10 times lookups for uniform random nodes by each node (in total $10N$ times lookups). In both experiments, FRT-Skip Graph kept path length drastically shorter than Skip Graph. The reason we consider to be is the number of effective entries and target nodes to select for shortcut links. In Skip Graph, as referred in Sec. II-A, a same node may be inserted two or more entries. Therefore, the number of effective entries in a routing table is often less than $\log N$. In contrast, in FRT-Skip Graph, each node can insert different nodes from each other into $L = \log N$ entries. Moreover, in Skip Graph, some entries at high level are often filled with closer nodes than their desirable nodes. Since FRT-Skip Graph can contain two or more nodes at a high level in a routing table and construct shortcut links to distant nodes, the path length seems to have been kept short.

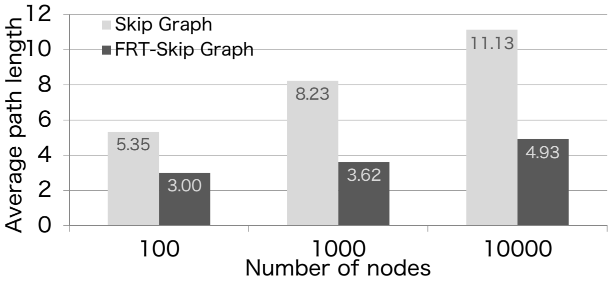

Fig. 4: Average path length.

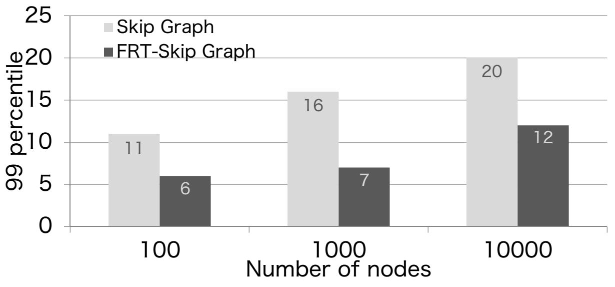

Fig. 5: 99 percentile of path length.

B. *Routing Table Size and Path Length*

We experimented to measure average path length while resizing a routing table by utilizing the feature of FRT-Skip Graph; dynamic routing table. In this experiment, we specified the number of nodes to $N = 10000$ and resized routing table size from $L = 14$ to $20, 40, 80$, and measured path length.

Figure 6 shows the result of this experiment. The averages of path length were calculated from 10 times lookups for uniform random nodes by each node (in total 10^5 times lookups). From this result, we find that average path length became shorter as routing table size increased. Especially, we resizing the size from $L = 14$ to $20, 40$, average path length was reduced significantly. Taking notice the values between $L = 40$ and 80, we find that it was little difference; however, it is because that improvement of a routing table was not enough. If the improvement continues by more times than 200, path length in $L = 80$ becomes shorter.

In this experiment, we showed the result in the situation on a large scale, such as $N = 10000$. On the other hand, we confirmed that in the situation on a small scale, such as $N = 100$, FRT-Skip Graph decreased path length to 1 by specifing routing table size to $L > N$. That is, we revealed that FRT-Skip Graph can handle both single-hop routing and multi-hop routing.

V. Extension of FRT-Skip Graph

In this section, we discuss several methods to improve the efficiency to construct a routing table by extend the design of FRT-Skip Graph.

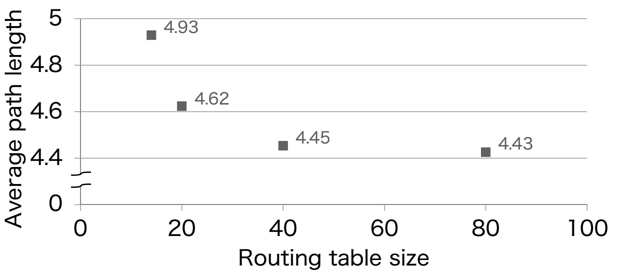

Fig. 6: Routing table size and average path length.

A. Adapting to Node Position

Since a network topology of Skip Graph is based on a linked list, contrary to a ring topology, there is heterogeneousness in routing performance brought by the position of a node. For example, we taking notice of a node at the forward side of the list, the number of former nodes in the list is less than the other. In this case, path length to backward nodes becomes longer than the average. On the other hand, at a node around the center of the list, path length to both directions is still short.

FRT-Skip Graph can deal with this heterogeneousness by adjusting the balance between two routing tables in both directions depending on the position of the node. In particular, a node autonomously reassigns routing table size—in this paper, it is L in each routing table, and the total is $2L$—to increase the number of entries in the routing table for the direction where a lot of nodes exist. To implement such an algorithm, we have to bring a method to estimate the node position. Alternatively, it is possible to redesign $\leq_{\mathrm{FRT\text{-}SG}}$ to manage shortcut links for both directions in only one routing table.

B. Efficient Entry Learning

Structured overlays based on FRT insert nodes' information into a routing table continually brought from various communication results. In FRT-Skip Graph, since it refines a routing table by referring levels of entries, it is significant to notice desirable nodes for a routing table; especially, the nodes at high levels.

Iterative forwarding (as shown in Fig. 3) that enables the querying node to notice relaying nodes is especially effective in FRT-Skip Graph to find nodes at high levels. This is because the relaying nodes select a next node that has long common prefixes in MVs, and therefore, the each node in route has a MV that has long common prefixes. For this reason, the initiating node can learn effectively desirable nodes by iterative forwarding.

In addition to entry learning with lookups for other nodes, we can improve the performance to construct a routing table by implementing a Skip Graph-style joining algorithm that a joining node fills its routing table with existing nodes.

VI. Conclusion

In this paper, we proposed FRT-Skip Graph, a novel structured overlay that can perform range queries similar to Skip Graph. FRT-Skip Graph solves the problem of Skip Graph that the construction of a routing table based on random digits brings useless entries and keeps path length drastically shorter than Skip Graph.

Since FRT-Skip Graph is designed based on FRT, it provides desirable features, that is, dynamic routing table size and high extensibility. Utilizing these features, for example, FRT-Skip Graph can decrease path length still shorter by maintaining more shortcut links and adapt to the requirements from applications by extending its construction of a routing table to utilize its entries effectively.

We revealed the routing performance of FRT-Skip Graph by the logical and mathematical proof, and the experimental results showed that it keeps path length much shorter than Skip Graph.

Future work includes an algorithm to adjust the balance between two routing tables in both directions depending on the position of the node. Besides, we will investigate behavior of FRT-Skip Graph with extremely small routing table size.

Acknowledgment

We wish to thank Prof. Kota Abe at Osaka City University for discussions and comments. This work was supported by JSPS KAKENHI Grant Numbers 25700008 and 26540161.

References

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[2] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Proc. IPTPS'02*. Springer, 2002, pp. 53–65.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. Middleware 2001*. Springer, 2001, pp. 329–350.

[4] J. Aspnes and G. Shah, "Skip Graphs," *ACM Transactions on Algorithms*, vol. 3, no. 4, Article 37, 2007.

[5] W. Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[6] 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 2011*. IEEE, 2011, pp. 72–81.

[7] T. Miyao, H. Nagao, and K. Shudo, "A Method for Designing Proximity-aware Routing Algorithms for Structured Overlays," in *Proc. IEEE ISCC 2013*. IEEE, 2013, pp. 508–514.

[8] H. Nagao and K. Shudo, "GFRT-Chord: Flexible Structured Overlay Using Node Groups," in *Proc. IEEE UIC 2014*. IEEE, 2014, pp. 187–195.

[9] Y. Ando, H. Nagao, T. Miyao, and K. Shudo, "Routing Table Construction Method Solely Based on Query Flows for Structured Overlays," in *Proc. IEEE P2P 2014*. IEEE, 2014, pp. 1–5.

[10] Y. Ando, H. Nagao, T. Miyao, and K. Shudo, "FRT-2-Chord: A DHT Supporting Seamless Transition between On-hop and Multi-hop Lookups with Symmetric Routing Table," in *Proc. ICOIN 2014*. IEEE, 2014, pp. 170–175.

[11] T. Miyao, H. Nagao, and K. Shudo, "A Structured Overlay for Non-uniform Node Identifier Distribution Based on Flexible Routing Tables," in *Proc. IEEE ISCC 2014*. IEEE, 2014.

[12] M. Hojo, H. Nagao, T. Miyao, and K. Shudo, "A Two-dimensional Structured Overlay Based on Flexible Routing Tables," in *Proc. IEEE ISCC 2015*. IEEE, 2015, pp. 309–314.

[13] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay Weaver: An Overlay Construction Toolkit," *Computer Communications*, vol. 31, no. 2, pp. 402–412, 2008.

[14] K. Shudo, "Overlay Weaver," http://overlayweaver.sourceforge.net, [Accessed 27 February 2016].