IEICE TRANS. INF. & SYST., VOL.E103–D, NO.3 MARCH 2020

PAPER Special Section on Foundations of Computer Science —Frontiers of Theory of Computation and Algorithm—

An Efficient Routing Method for Range Queries in Skip Graph

Ryohei BANNO[†a)] ***and*** **Kazuyuki SHUDO**[†], ***Members***

SUMMARY Skip Graph is a promising distributed data structure for large scale systems and known for its capability of range queries. Although several methods of routing range queries in Skip Graph have been proposed, they have inefficiencies such as a long path length or a large number of messages. In this paper, we propose a novel routing method for range queries named Split-Forward Broadcasting (SFB). SFB introduces a divide-and-conquer approach, enabling nodes to make full use of their routing tables to forward a range query. It brings about a shorter average path length than existing methods, as well as a smaller number of messages by avoiding duplicate transmission. We clarify the characteristics and effectiveness of SFB through both analytical and experimental comparisons. The results show that SFB can reduce the average path length roughly 30% or more compared with a state-of-the-art method.

key words: *Skip Graph, overlay networks, peer-to-peer networks, routing algorithms, distributed algorithms*

1. Introduction

Structured overlay networks are popular techniques to provide superior scalability and robustness for large scale distributed systems, such as video streaming [1], pub/sub messaging [2], Blockchain [3], and IoT [4].

Skip Graph [5] is one of the structured overlay networks, whose remarkable feature is the capability of range queries. Different from the well-known DHTs [6]–[8], Skip Graph does not hash keys so that the order of them is preserved. This enables a node in Skip Graph to issue a query with specifying a range, and to find nodes having keys within it.

Although there are some existing methods of routing range queries [9], [10], they have inefficiencies such as involving a large number of messages by duplicate transmission or a high latency by a long path length.

In this paper, we propose a novel routing method for range queries in Skip Graph, named Split-Forward Broadcasting (SFB). SFB utilizes a divide-and-conquer approach as like a state-of-the-art method [10], but has a different dividing policy. In SFB, each node which receives a range query divides the target range into subranges by keys of its neighbor nodes, and delegates the subranges to them. By using SFB, a shorter average path length than existing methods can be achieved, as well as avoiding duplicate transmission.

In addition, we provide pragmatic knowledge of tuning the parameters in Skip Graph, e.g., the size of the alphabet for membership vectors. Since such parameters are considered to affect the efficiency of routing range queries, we make their influence clear by simulation experiments.

The contributions of this paper are threefold:

- First, we present a novel routing method for range queries in Skip Graph.
- Second, we clarify the superiority of the proposed method through both analytical and experimental comparisons with existing methods.
- Third, by simulation experiments, we clarify the influence of the parameters in Skip Graph on the efficiency of routing range queries.

This paper is an extended version of [11], which reported basic idea and restrictive analytical discussion. The differences include a detailed algorithm with a pseudo code (Sect. 3), a proof of equivalence between the delivery tree in SFB and a binomial tree (Sect. 4), and evaluation by simulation experiments (Sect. 5). Furthermore, this paper clarifies the influence of parameters in Skip Graph on routing range queries (Sect. 5.2).

The rest of this paper is organized as follows: Sect. 2 introduces related studies of handling range queries in Skip Graph. Section 3 explains the proposed method, while the analytical comparison with existing methods is described in Sect. 4. Section 5 presents experimental evaluation, including the discussion on the influence of the parameters in Skip Graph. Finally, we conclude this paper in Sect. 6.

2. Related Work

Skip Graph [5] is a kind of structured overlay networks, providing the capability of range queries. Each node has a key and a random sequence called a membership vector, which consists of symbols contained in a finite alphabet Σ. A node can issue a query with specifying a target range or a value in the key space. The query is delivered to nodes whose keys are included in the range or exactly matched with the value.

Skip Graph composes a multiplex structure of a skip list [12]. Figure 1 shows an example of Skip Graph with $\Sigma = \{0, 1\}$. Level 0 is a doubly linked list that consists of all nodes sorted in the order of keys. In level i, each node composes a doubly linked list with nodes whose membership vectors have the same first i digits.

When a node issues an exact-match query, the search process starts from the highest level of the node. The query

Manuscript received April 26, 2019.
Manuscript revised September 26, 2019.
Manuscript publicized December 9, 2019.

[†]The authors are with Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
a) E-mail: banno@computer.org

DOI: 10.1587/transinf.2019FCP0008

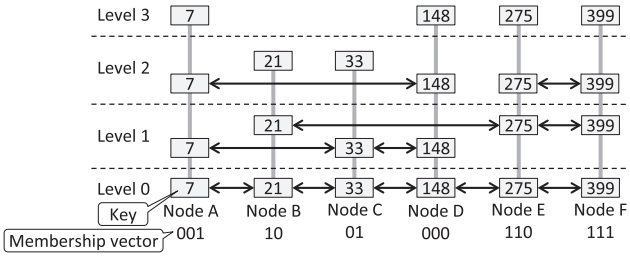

Fig. 1 Example of Skip Graph.

is forwarded among nodes in the same manner as a skip list, i.e., it skips long distance at the higher level and gradually moves down to level 0. The path length of forwarding the query is $O(\log N)$ in Skip Graph of N nodes, while the size of the routing table on each node is also $O(\log N)$.

2.1 Routing Range Queries in Skip Graph

In Skip Graph, a range query is first delivered to one of the nodes within the target range by the previously described way of exact-match queries. Subsequently, the query is spread from the one to other nodes within the range. There are several existing methods of spreading queries. Beltran et al. [9] discussed the following methods with respect to the number of messages and the path length.

- Sequential traversing
  A query is forwarded along the doubly linked list at level 0, until the upper or lower bound of the range is found.
- Broadcasting without memory
  Each node received a query forwards it to all its neighbors within the range. Each node also keeps track of the query already received to discard duplicate query.
- Broadcasting with memory
  It is a variation of the above broadcasting method. When each node forwards the query, it includes a list of nodes, which have already received the query, into the message. This reduces the number of messages partly by avoiding some of duplicate transmission.

Note that another method called Tree-based in [9] is out of the scope of this paper. This is because it uses additional links, being particular to Skip Tree Graph [13] and requiring a commensurate maintenance cost; we are focusing on handling range queries in a normal topology of Skip Graph for versatility.

The sequential traversing method can deliver a range query to all corresponding nodes with the minimum number of messages, without any duplicate transmission. However, it requires long path lengths, especially when the number of nodes within the range is large. Contrary, broadcast-style methods outperform the sequential traversing method in terms of the path length, but they involve a large number of messages by duplicate transmission. Even in the broadcasting with memory method, the experimental results in [9] show that the number of messages is quite larger than in the sequential traversing method; it cannot eliminate duplicate transmission completely. That is, there is a difficulty of suppressing both the path length and the number of messages by using these methods.

Algorithm 1: Routing process in MRF.

```
1  uponReceiving(range, query)
2      leftDelegationNode ← NULL;
3      rightDelegationNode ← NULL;
4      leftMaxLevel ← −1;
5      rightMaxLevel ← −1;
6      foreach neighbor ∈ routingTable do
7          if neighbor.key is within range then
8              if neighbor.key < key and neighbor.level >
                 leftMaxLevel then
9                  leftMaxLevel ← neighbor.level;
10                 leftDelegationNode ← neighbor;
11             if neighbor.key > key and neighbor.level >
                 rightMaxLevel then
12                 rightMaxLevel ← neighbor.level;
13                 rightDelegationNode ← neighbor;
14     if leftDelegationNode ≠ NULL then
15         leftSubRange ← clone of range;
16         leftSubRange.setRightOpenBound(key);
17         Send leftSubRange and query to leftDelegationNode;
18     if rightDelegationNode ≠ NULL then
19         rightSubRange ← clone of range;
20         rightSubRange.setLeftOpenBound(key);
21         Send rightSubRange and query to
             rightDelegationNode;
```

There exists another state-of-the-art method, called Multi-Range Forwarding (MRF) [10], which can improve the above difficulty to some extent. MRF introduces a divide-and-conquer approach. A query with its target range R is forwarded as follows: when a node whose key is within R receives the query, the node divides R into subranges by its key. Then the node chooses a neighbor node for each subrange, which is the neighbor of the highest level among neighbors within the subrange. Subsequently, the query attached with the subrange instead of R is forwarded to the neighbor node. By recursively forwarding the query in the same manner, it is eventually delivered to all nodes having keys within R.

The pseudo-code of MRF is shown in Algorithm 1. The function **uponReceiving** is called when a node receives a query with a target range containing its key. At first, the node tries to extract neighbor nodes to which subranges are delegated for each of the left side and the right side. (lines 2-13). If a corresponding node exists, the query is forwarded with a subrange to it (lines 14-21). In the pseudo-code, The functions **setRightOpenBound** and **setLeftOpenBound** change the bound of a range to the value specified in the argument as an open bound, i.e., the bound does not include the endpoint.

Figure 2 shows an example of routing with the use of MRF. The blue rounded rectangles indicate the target range and its subranges. When node A receives a query with its tar-

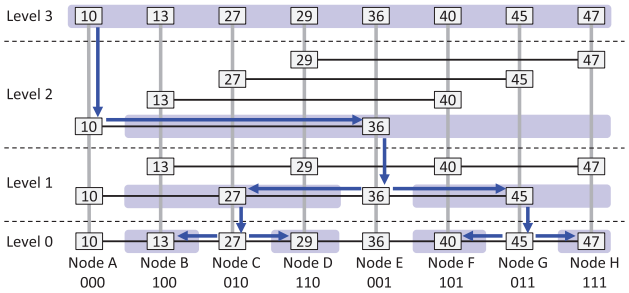

Fig. 2 Example of routing with MRF.

get range $R = [8, 50]$ for instance, the range is divided by the key 10 into two subranges: $[8, 10)$ and $(10, 50]$. Among the neighbors of node A within $(10, 50]$, node E is the neighbor of the highest level. Therefore, node A forwards the query with the subrange $(10, 50]$ to node E. Subsequently, node E forwards the query to node C and node G in the same manner, with the subrange $(10, 36)$ and the subrange $(36, 50]$ respectively. Node C also forwards it to node B and node D, while node G forwards to node F and node H.

MRF can avoid duplicate transmission, i.e., each node within the target range receives the query only once. In addition, it achieves shorter path lengths than the sequential traversing method, owing to that each node forwards the query to not one but at most two neighbor nodes.

Although MRF outperforms other existing methods in terms of both the path length and the number of messages, the former still tends to be long. For example, the path length from node A to node B in Fig. 2 is 3, despite node B has a direct link with node A at level 0. Such inefficiency is caused by the structure of the delivery tree. As shown in the figure, MRF forms a delivery tree similar to a binary tree, so that almost half of nodes receive the query at the leaves of the tree with the maximum path length.

We describe analytical comparison of these existing methods in detail, later in Sect. 4.

3. Split-Forward Broadcasting

We propose a novel routing method for range queries named Split-Forward Broadcasting (SFB), which can reduce the average path length of MRF.

SFB utilizes a divide-and-conquer approach as like MRF, but has a different dividing policy. In SFB, each node which receives a range query divides the target range into subranges by keys of its neighbor nodes, whereas in MRF the range is divided by the key of itself. Namely, a query with its target range R is forwarded as follows: when a node whose key is within R receives the query, the node first divides R into two subranges by its key, i.e., the left side and the right side. After that, the node proceeds a same process for each of them. It divides the subrange into smaller subranges by using the keys of all its neighbor nodes placed within the subrange. Then the node forwards the query to each of them with the corresponding subrange, instead of R. By recursively forwarding the query in the same manner, the query is eventually delivered to all nodes which have keys within R.

Algorithm 2: Routing process in SFB.

```
1  uponReceivingAtStart(range, query)
2      leftSubRange ← clone of range;
3      leftSubRange.setRightClosedBound(key);
4      rightSubRange ← clone of range;
5      rightSubRange.setLeftClosedBound(key);
6      uponReceiving(leftSubRange, query, FALSE);
7      uponReceiving(rightSubRange, query, TRUE);
8  uponReceiving(range, query, isRightSide)
9      delegationNode ← NULL;
10     neighbors ← NULL;
11     if isRightSide = TRUE then
12         neighbors ← routingTable.getRightNeighbors();
13     else
14         neighbors ← routingTable.getLeftNeighbors();
15     maxLevel ← −1;
16     foreach neighbor ∈ neighbors do
17         if neighbor.key is within range and neighbor.level >
             maxLevel then
18             maxLevel ← neighbor.level;
19             delegationNode ← neighbor;
20     if delegationNode ≠ NULL then
21         subRange ← clone of range;
22         if isRightSide = TRUE then
23             subRange.setLeftClosedBound(delegationNode.key);
24             range.setRightOpenBound(delegationNode.key);
25         else
26             subRange.setRightClosedBound(delegationNode.key);
27             range.setLeftOpenBound(delegationNode.key);
28         Send subRange, query and isRightSide to
             delegationNode;
29         uponReceiving(range, query, isRightSide);
```

The pseudo-code of SFB is shown in Algorithm 2. The function **uponReceivingAtStart** is called only when a range query is received by the first node in the target range. When the secondary or succeeding nodes within the range receive the query, **uponReceiving** is called instead of **uponReceivingAtStart**.

In **uponReceivingAtStart**, at first, the node divides the range into two subranges by its key (lines 2-5). Subsequently, the node calls the function **uponReceiving** for each of the left side and the right side (lines 6-7). In **uponReceiving**, the node tries to extract a delegation node, which is a neighbor on the specified side and connected at the highest level among neighbors within the range (lines 9-19). If a corresponding node exists, the node divides the range into two subranges by the key of the delegation node. The one farther from the key of the node is attached to the query, and then it is forwarded to the delegation node. Another subrange, which contains the key of the node, is used in the recursive calling of **uponReceiving** (line 29). In the pseudo-code, the functions **setRightClosedBound** and **setLeftClosedBound** change the bound of a range to the

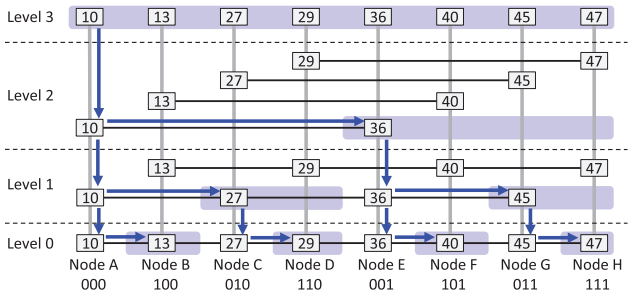

Fig. 3 Example of routing with SFB.

value in the argument as a closed bound, i.e., the bound includes the endpoint.

Figure 3 shows an example of routing with the use of SFB. Same as Fig. 2, The blue rounded rectangles indicate the target range and its subranges. When node A receives a query with its target range $R = [8, 50]$, the range is divided into two subranges: $[8, 10]$ and $[10, 50]$. Among the neighbors of node A within $[10, 50]$, node E is the neighbor of the highest level. Therefore, node A divides $[10, 50]$ into $[10, 36)$ and $[36, 50]$ by the key of node E, then forwards the query with the subrange $[36, 50]$ to node E. Subsequently, node A performs the same process for the remaining subrange $[10, 36)$ at lower levels recursively. That is, node A forwards the query with the subrange $[27, 36)$ to node C, and then forwards the query with the subrange $[13, 27)$ to node B. Each node which receives the query from node A forwards the query to its neighbors in the same manner, so that the query is eventually delivered to all nodes having keys within the target range.

SFB enables nodes to make full use of their routing tables to forward a range query, unlike MRF. For example, node A in Fig. 3 forwards the query to three neighbor nodes, whereas in MRF each node forwards to only at most two neighbors. Therefore, SFB achieves a shorter average path length than MRF as well as avoiding duplicate transmission.

4. Analytical Comparison

We give analytical comparison of SFB with the methods described in Sect. 2, from the viewpoints of the path length and the number of messages required inside target ranges.

Table 1 shows the summary of differences, where N_R denotes the number of nodes within a target range. For example, in the case of Fig. 2 and 3, N_R is equal to 8.

Regarding the sequential traversing method, a query is forwarded along the level 0 list. In the case of Fig. 2 and 3, the query issued by node A is forwarded in the order of B, C, D, E, F, G, H. Accordingly, both the path length and the number of messages are 7. That is, the path length and the number of messages in the sequential traversing method are linear with respect to N_R.

On the other hand, both broadcasting methods require only a logarithmic path length. This relies on the nature of the topology of Skip Graph, since broadcasting methods basically achieve the shortest path length. Note that we

Table 1 Comparison of routing methods.

	Path length	# of messages
Sequential traversing	$O(N_R)$	$O(N_R)$
Broadcasting with/without memory	$O(\log N_R)$	$O(N_R \log N_R)$
MRF	$O(\log N_R)$	$O(N_R)$
SFB	$O(\log N_R)$	$O(N_R)$

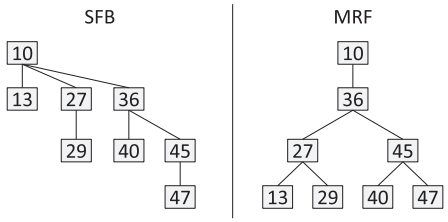

Fig. 4 Difference of tree structures.

are focusing on the generic characteristics, even though the path length is affected by the communication latency among nodes. In regard to the number of messages, these methods require $O(N_R \log N_R)$ messages. This is because each of N_R nodes has $O(\log N_R)$ neighbors inside the target range and they forward a query to all their neighbors within the target range except for the sender node.

As for MRF and SFB, a query is forwarded by gradually moving down to lower levels. Namely, the path length becomes linear with respect to the highest level where the start node has a neighbor within the target range. Since the highest level is logarithmic of N_R, the path length is $O(\log N_R)$. In addition, they require only $O(N_R)$ messages, because each node within the target range receives the same query only once. Consequently, they have advantages of both the sequential traversing method and the broadcasting methods.

4.1 Difference in Tree Structures

Although MRF and SFB have basically the same tendencies in regard to the path length and the number of messages, there is an inefficiency that the actual path length in MRF tends to be long as previously mentioned in Sect. 2. We discuss the difference about the actual path length from the viewpoint of the structure of delivery trees.

Figure 4 shows the difference of tree structures in the case of the examples in Fig. 2 and Fig. 3. Note that these examples are using a balanced Skip Graph topology to simplify, i.e., every link in level i skips exactly $|\Sigma|^i - 1$ nodes. In the examples, the size of the alphabet for membership vectors is $|\Sigma| = 2$. The principle difference between SFB and MRF is that SFB enables each node to forward the query to all possible neighbors, whereas in MRF each node forwards to at most two neighbors. As a result, SFB composes a tree as shown in the left side of Fig. 4, while MRF composes a binary tree-like structure as shown in the right side.

The tree structure of SFB is known as a binomial tree, which is used in a binomial heap [14]. Binomial trees are

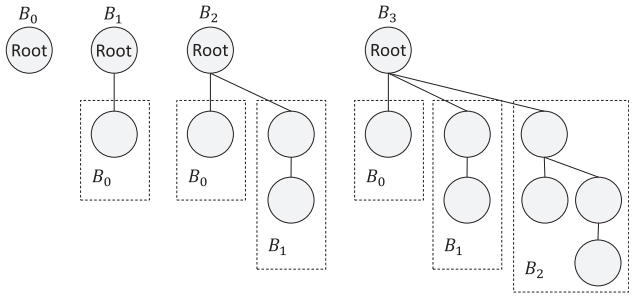

Fig. 5 Recursive structure of binomial trees.

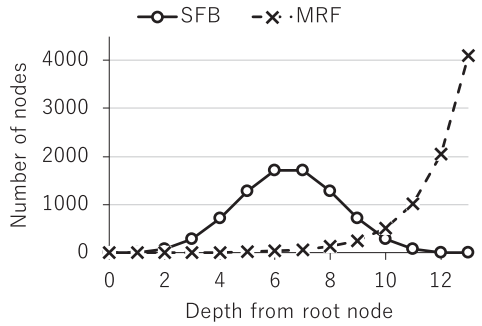

Fig. 6 Comparison of number of nodes at each depth.

defined recursively as shown in Fig. 5. A binomial tree of order 0, denoted as B_0, is a single node. A binomial tree of order p, denoted as B_p, has a root node whose children are roots of $B_{p-1}, B_{p-2}, \cdots, B_0$.

In the following discussion in this section, we use the assumptions below to prove and analyze the characteristics of SFB and MRF:

- $|\Sigma| = 2$.
- A completely balanced topology is composed.
- The number of nodes within a target range is 2^r.
- A range query is first received by the leftmost node in its target range.

We also use the following notations: n_i denotes the ith node from the leftmost node within the target range, $K(n_i)$ denotes the key of n_i, and L_j denotes level j in Skip Graph.

The target range can be denoted as $[K(n_0), K(n_{2^r}))$ for whatever the original range is, because the leftmost node and the rightmost node are n_0 and n_{2^r-1} respectively. Here, we have the following lemmas:

Lemma 1. *The node which has a link with n_0 at the highest level within the range is $n_{2^{(r-1)}}$, where the level is L_{r-1}.*

Proof. n_0 has a link with n_{2^i} at L_i, clearly from the assumptions of the completely balanced topology and $|\Sigma| = 2$. Since the rightmost node within the range is n_{2^r-1}, the highest level L_j at which n_0 has a link within the range satisfies $2^j \leq 2^r - 1$. Consequently, the largest j satisfying the condition is $r - 1$. □

Lemma 2. *In using SFB, n_0 forwards the query to n_{2^i} at L_i, for each i from $r-1$ to 0. The subrange attached to the query becomes $[K(n_{2^i}), K(n_{2^{(i+1)}}))$.*

Proof. From Lemma 1, n_0 first forwards the query to $n_{2^{(r-1)}}$. After that, the remaining subrange is $[K(n_0), K(n_{2^{(r-1)}}))$. The number of nodes contained in this subrange is 2^{r-1}. Therefore, we can apply Lemma 1 recursively by replacing r to $r - 1$. This means that n_0 forwards the query to n_{2^i} for each i from $r - 1$ to 0. Here, the subrange forwarded from n_0 to $n_{2^{(r-1)}}$ is obviously $[K(n_{2^{(r-1)}}), K(n_{2^r}))$, while the subrange forwarded from n_0 to $n_{2^{(r-2)}}$ is $[K(n_{2^{(r-2)}}), K(n_{2^{(r-1)}}))$. Consequently, by also considering recursively, the subrange for each i becomes $[K(n_{2^i}), K(n_{2^{(i+1)}}))$. □

From these lemmas, now we have the following theorem:

Theorem 1. *SFB forms a binomial tree in a balanced Skip Graph topology where $|\Sigma| = 2$.*

Proof. In Lemma 2, n_{2^i} is placed at the leftmost of the subrange which it receives. In addition, the subrange contains 2^i nodes. Hence, we can apply Lemma 2 recursively by replacing 2^r to 2^i with considering n_{2^i} as new n_0. Namely, when a range contains 2^j nodes, the delivery tree starting from the leftmost node of the range has always the same structure for same j. Let T_j denote this tree structure. Here, clearly from Lemma 2, the child nodes of the root node of T_r are the root nodes of $T_{r-1}, T_{r-2}, \cdots, T_0$. Since T_0 is equivalent to a single node, the tree composed by SFB coincides with a binomial tree. □

A binomial tree has a feature of that the number of nodes at each depth of the tree is the same as binomial coefficient; there are $\binom{p}{d}$ nodes at the depth d in B_p [14]. On the other hand, the number in MRF is exponentiation of 2 due to its binary tree structure.

When we assume the number of nodes within the target range is $2^{13} = 8{,}192$, the difference of the number of nodes at each depth between SFB and MRF becomes as shown in Fig. 6. Regarding MRF, the number of nodes at the deepest depth 13 is the largest. Unlike this, the tree of SFB has the largest number of nodes at the depth 6 and 7. In other words, more nodes have shorter path lengths in SFB than in MRF. This difference makes the superiority of SFB regarding the average path length.

4.2 Average Path Length

By considering the feature of binomial trees, the tree of SFB has $\binom{\log N_R}{d}$ nodes at the depth d ($d = 0, 1, \ldots, \log N_R$). Since the depth is equivalent to the path length from the root node to the corresponding node, we can calculate the average path length of SFB, L_{SFB}, as follows:

$$L_{SFB} = \frac{1}{N_R} \sum_{d=0}^{\log N_R} \left\{ \binom{\log N_R}{d} \cdot d \right\}$$

$$
\begin{aligned}
&= \frac{1}{N_R} \sum_{d=1}^{\log N_R} \left\{ \binom{\log N_R}{d} \cdot d \right\} \\
&= \frac{1}{N_R} \sum_{d=1}^{\log N_R} \left\{ \frac{\log N_R}{d} \cdot \binom{\log N_R - 1}{d-1} \cdot d \right\} \\
&= \frac{\log N_R}{N_R} \sum_{d=1}^{\log N_R} \binom{\log N_R - 1}{d-1} \\
&= \frac{\log N_R}{N_R} \sum_{d-1=0}^{\log N_R - 1} \binom{\log N_R - 1}{d-1} \\
&= \frac{\log N_R}{N_R} \cdot 2^{\log N_R - 1} \\
&= \frac{\log N_R}{2}
\end{aligned}
$$

For the tree of MRF, it has 2^{d-1} nodes at the depth d ($d = 1, 2, \ldots, \log N_R$). Therefore, the average path length L_{MRF} can be calculated as follows:

$$
L_{MRF} = \frac{1}{N_R} \sum_{d=0}^{\log N_R - 1} \left\{ (d+1) \cdot 2^d \right\}
$$

Let m and S denote:

$$
\begin{aligned}
m &= \log N_R \\
S &= \sum_{d=0}^{m-1} \left\{ (d+1) \cdot 2^d \right\}
\end{aligned}
$$

Here, we can find

$$
\begin{aligned}
S &= 1 \cdot 2^0 + 2 \cdot 2^1 + \cdots + m \cdot 2^{m-1} \\
2S &= 1 \cdot 2^1 + 2 \cdot 2^2 + \cdots + m \cdot 2^m
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
S &= 2S - S \\
&= m \cdot 2^m - (2^0 + 2^1 + \cdots + 2^{m-1}) \\
&= m \cdot 2^m - (2^m - 1) \\
&= (m-1) \cdot 2^m + 1 \\
&= (\log N_R - 1) \cdot N_R + 1
\end{aligned}
$$

$$
\begin{aligned}
\therefore L_{MRF} &= \frac{S}{N_R} \\
&= \log N_R - 1 + \frac{1}{N_R}
\end{aligned}
$$

Figure 7 illustrates values of L_{SFB} and L_{MRF}, where the horizontal axis represents N_R. From these, we can say that the average path length in SFB is approximately half of that in MRF. Reducing the average path length is quite effective for improving the latency performance. Even though some of the nodes have a larger number of child nodes in SFB than in MRF, the former is considered to be superior because communication delay is generally much more dominant than processing delay. Note that the maximum number of child nodes in SFB is not so large, since it is bounded by $O(\log N_R)$. Furthermore, the shorter average path length also brings about a higher fault-tolerance, by reducing the probability of being affected by the failure of nodes. For example, the average number of descendant nodes of each node on the trees in Fig. 4 are 1.5 for SFB and 2.125 for MRF. By such a smaller number of descendant nodes in SFB, it is expected to lead the smaller influence by nodes' failure.

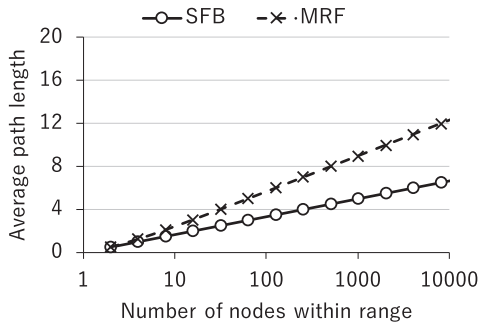

Fig. 7 Comparison of average path length in balanced Skip Graph.

As described above, the discussion in this section assumes the completely balanced topology of Skip Graph. Therefore, it shows somewhat different results from the experiments with randomly generated membership vectors, although assuming the balanced topology is helpful for understanding the characteristics analytically. The difference and its meanings are discussed in Sect. 5.

5. Evaluation

We evaluate SFB through experiments with a simulation program implemented in Java (Java SE Development Kit 8). The simulator is operated on a machine with a quad-core CPU (Intel Core i7-8650U 1.90GHz), 16GB memory, and Windows 10 operating system.

In this evaluation, our simulator generates 10,000 nodes to compose Skip Graph with $|\Sigma| = 2$[†]. Subsequently, it measures the average path length for SFB and MRF respectively, while changing the number of nodes within the target range (N_R). Each measurement is conducted with the following steps:

1. A range query is issued by the leftmost node in its range[††].

[†]We use $|\Sigma| = 2$ which is the most commonly used value when discussing and implementing Skip Graph. For example, [5] discusses some sections with an assumption of $|\Sigma| = 2$.

[††]Starting from the leftmost node within the range is one of the choices in actual use. In Skip Graph, routing toward the ends of a range is capable, whereas that toward a specific position within the range such as the 100th node from the leftmost node is difficult. Therefore, practical choices for the position of the start node within the range are the leftmost node, the rightmost node, and any node which receives the query firstly within the range. Since the node which receives the query firstly within the range is not always the same in actual use except for the case using the endmost nodes, we use the leftmost node as the fixable start position within the range.

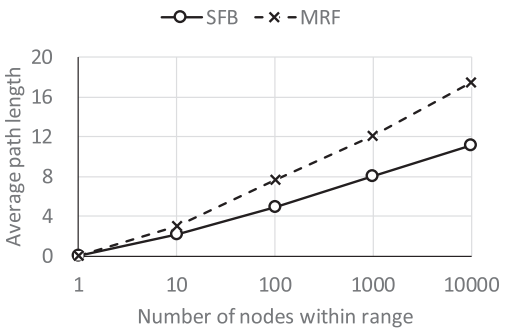

Fig. 8 Average path length of SFB and MRF.

2. The query is forwarded by using SFB or MRF, with keeping track of the path length from the start node to each node within the target range.
3. After completion of delivering, the simulator calculates the average path length.
4. The above steps are repeatedly conducted five times, and finally calculates the average of the five times.

Note that we fixed the number of nodes, 10,000, through experiments in this section because it does not affect the result. This can be said from the discussion of analytical comparison in Sect. 4; the path length of both SFB and MRF depend on N_R but not the number of whole nodes.

Figure 8 shows the result. Although SFB and MRF have basically the same tendencies in response to N_R, SFB achieves shorter average path lengths. In the figure, SFB reduces the average path lengths of MRF by 26.0% (N_R = 10), 35.23% (N_R = 100), 33.19% (N_R = 1,000), and 35.96% (N_R = 10,000) respectively.

Different from using the balanced topology discussed in Sect. 4.2, the reduction rate by using SFB does not come up to around 50%. From Fig. 7 and 8, average path lengths for both SFB and MRF in the experiment are longer than those derived from the analytical comparison. Thereby, the reduction rate by using SFB becomes lower than 50%. This difference between the experimental results and the analytical discussion is because the topology in the experiment is not completely balanced due to randomly generated membership vectors.

The fact that actual performance becomes lower than the theoretical value is representing one of the shortcomings of randomized algorithms like Skip Graph. Namely, the performance of such randomized algorithms is affected by the deviation of random numbers. To overcome this defect, methods to avoid or improve the influence of random number generation exist: Deterministic SkipNet [15] is a method for constructing a scalable overlay network which is similar to Skip Graph but is not relying on randomized numbers. We can avoid the influence of random number generation by using this deterministic algorithm. Self-Refining Skip Graph [16] is another approach. It aims at refining the topology of Skip Graph dynamically towards the completely balanced one. By using Self-Refining Skip Graph, we can obtain the performance nearly the theoretical value since the topology becomes approximately balanced after a sufficient period of time to refine. Note that these methods take more maintenance cost, e.g., the number of messages required for nodes' joining/leaving or for refining the topology, than Skip Graph.

From these, we can say that the analytical discussion in Sect. 4 indicates the performance in an ideal topology of Skip Graph and we can obtain it by using some methods [15], [16] in exchange for more maintenance cost. On the other, experimental results in this section indicate the performance when we use Skip Graph as it is.

5.1 Discussion on Latency

As described in Sect. 4, reducing the average path length is effective for better latency. Primary factors of latency for delivering a range query are processing delay in each intermediate node on the path and communication delay between nodes. Reducing the average path length leads to suppressing the latter which is more dominant in general.

Reducing the average path length is particularly effective in the case of that nodes are scattered over a wide area rather than placed inside a local area network such as a data center. For example, some existing studies [2], [17] using range queries of Skip Graph assume that a large number of nodes are placed over a wide area, at the network edge, and provide the functionality of publish/subscribe messaging for IoT devices. In such cases, the communication delay between nodes tends to be quite large. Indeed, according to the Global Ping Statistics provided by WonderNetwork [18], round trip times (RTTs) among cities in the world reach as large as hundreds of milliseconds. Even inside one country, for example with Japan, RTTs among different cities are around 10 milliseconds or more. If we assume that communication delays between nodes are uniformly 10 or 100 milliseconds and the influence of processing delay is negligible, the differences in the average latencies between SFB and MRF become approximately 62.7 or 627 milliseconds respectively in the case of N_R = 10,000 in the experiment. This is based on the RTTs with PING and the differences could grow larger according to the size of each query in practice. Such differences are expected to improve the user experience (UX) in applications of Skip Graph[†].

Note that the above discussion is for the average path lengths, not for the maximum path lengths. The maximum path lengths, i.e., the maximum depth of delivery trees, are basically the same between SFB and MRF. Although the maximum path length is also important in consideration of latency, it is out of the scope of this paper and one of our future works. We will consider some approaches such as introducing the priority of receivers; managing receiver nodes so that they are placed at shallower depth on the delivery tree according to the requirement for the latency.

[†]It is said that even a relatively small delay, e.g., 10 to 100 milliseconds, highly affects UX [19], [20].

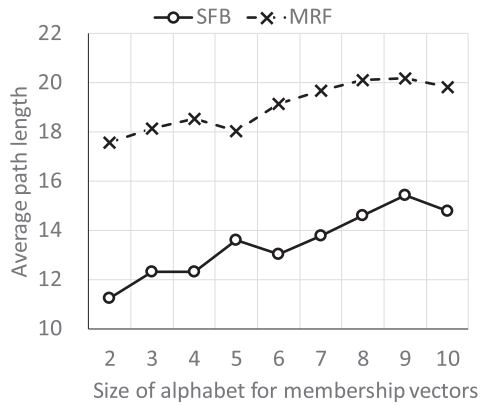

Fig. 9 Influence of alphabet size on average path length.

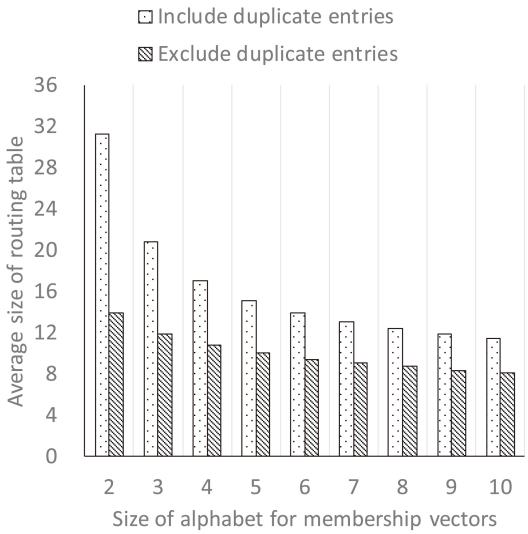

Fig. 10 Influence of alphabet size on routing table size.

5.2 Influence of Parameters in Skip Graph

As mentioned in Sect. 1, parameters in Skip Graph could affect the efficiency of routing range queries. In this section, we focus on the size of the alphabet for membership vectors and the position of the start node within the target range.

5.2.1 Size of Alphabet for Membership Vectors

We confirm the influence of $|\Sigma|$. In this experiment, our simulator generates 10,000 nodes to compose Skip Graph, and measures both the average path length and the average size of routing tables, while changing $|\Sigma|$. We measure two kinds of the size of routing tables: including duplicate entries, and excluding duplicate entries. The former is simply counting entries in routing tables, whereas the latter is counting different entries as 1 if they are a same node. N_R is set to 10,000[†], and each measurement is conducted five times repeatedly.

Figure 9 and Fig. 10 shows the result. The average size of routing tables seems to be smaller exponentially in response to the increase of $|\Sigma|$. We can also see that more than half of entries in routing tables are duplicate in the case of $|\Sigma| = 2$. The duplication ratio becomes lower by increasing $|\Sigma|$. This is because the larger the alphabet size becomes, the lower the probability of that two nodes connected at a level join the same list at a one higher level becomes.

On the other hand, the average path length is growing almost linearly in response to the increase of $|\Sigma|$. The tendencies of SFB and MRF seem to have no big difference.

From these, we can say that increasing $|\Sigma|$ is effective for reducing the size of routing tables especially when $|\Sigma|$ is relatively small. Increasing $|\Sigma|$ also leads the increase of the average path length, but the influence is limited. For example, when we use SFB and set $|\Sigma| = 10$, the average path length is still shorter than that of using MRF with $|\Sigma| = 2$.

[†]Note that $N_R = 10{,}000$ is a possible scale in actual use. For example, video streaming is one of the applications of structured overlay networks as described in Sect. 1. In such applications, N_R possibly corresponds to the number of concurrent viewers of a video streaming channel which could be 10,000 or more. Indeed, there is a record which has over 10 millions concurrent viewers [21].

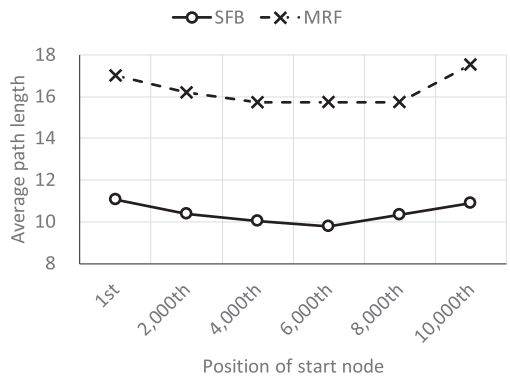

Fig. 11 Influence of position of start node.

5.2.2 Position of Start Node

We also confirm the influence of the position of the start node within the target range. As mentioned previously, a range query is forwarded to one of the nodes within the range by the way of routing exact-match queries. That is, we have options about where routing inside the range is started from; the leftmost node, the rightmost node, or any node which receives the query firstly within the range. These choices could affect the efficiency of routing range queries.

In this experiment, our simulator generates 10,000 nodes to compose Skip Graph with $|\Sigma| = 2$, and measures the average path length. Measurement is conducted with changing the start position between the 1st and the 10,000th, counting from the leftmost node within the range. N_R is set to 10,000, and each measurement is conducted five times repeatedly.

Figure 11 shows the result. In the case of starting from

the leftmost node or the rightmost node, the average path length tends to be longer than other positions. Although it is considered that the start position nearer the center of the range makes a shorter average path length, the difference between start positions other than both ends of the range seems to be small.

From the result, it is less efficient to choose the leftmost node or the rightmost node as the start node, unless there is any reason to process routing in one direction.

6. Conclusion

In this paper, we proposed SFB, which is a novel method for routing range queries in Skip Graph. SFB achieves a shorter path length and a smaller number of messages compared with existing methods. From the result of the simulation experiment, SFB can reduce the average path length roughly 30% or more than a state-of-the-art method MRF. As well as improving the latency performance, this shorter average path length is effective for obtaining a higher fault-tolerance by lowering the probability that queries are caught in failure of nodes.

We also discussed the influence of the parameters in Skip Graph. We clarified that increasing the alphabet size is effective for reducing the size of routing tables, though it leads the increase of the average path length to a rather small extent. As for the position of the start node, the result of the experiment showed that starting from the ends makes the average path length longer than other positions. These knowledges are helpful for users to design and implement Skip Graph-based systems.

Future work includes adaptive utilization of SFB and the other routing methods. Besides, we plan to clarify the applicability of SFB-style routing to other structured overlay networks, which are capable of range queries.

Acknowledgments

This work was supported by SECOM Science and Technology Foundation. This work was supported by JSPS KAKENHI Grant No.19K20253. This work was supported by New Energy and Industrial Technology Development Organization (NEDO). This work was supported by Suematsu Fund.

References

[1] N. Ramzan, H. Park, and E. Izquierdo, "Video streaming over P2P networks: Challenges and opportunities," Signal Processing: Image Communication, vol.27, no.5, pp.401–411, May 2012.
[2] Y. Teranishi, R. Banno, and T. Akiyama, "Scalable and locality-aware distributed topic-based pub/sub messaging for IoT," Proc. IEEE Global Communications Conference, pp.1–7, Dec. 2015.
[3] J. Chen, "Flowchain: A distributed ledger designed for peer-to-peer IoT networks and real-time data transactions," Proc. International Workshop on Linked Data and Distributed Ledgers, May 2017.
[4] D. Wu, D.I. Arkhipov, E. Asmare, Z. Qin, and J.A. McCann, "UbiFlow: Mobility management in urban-scale software defined IoT," Proc. IEEE Conference on Computer Communications, pp.208–216, April 2015.
[5] J. Aspnes and G. Shah, "Skip Graphs," ACM Transactions on Algorithms, vol.3, no.4, pp.37:1–37:25, Nov. 2007.
[6] I. Stoica, R. Morris, D. Karger, M.F. 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, Oct. 2001.
[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," Proc. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, pp.329–350, Nov. 2001.
[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," Proc. ACM conference on Applications, technologies, architectures, and protocols for computer communications, pp.161–172, Oct. 2001.
[9] A. González-Beltrán, P. Milligan, and P. Sage, "Range queries over Skip Tree Graphs," Computer Communications, vol.31, no.2, pp.358–374, Feb. 2008.
[10] Y. Konishi, M. Yoshida, S. Takeuchi, Y. Teranishi, K. Harumoto, and S. Shimojo, "An extension of Skip Graph to store multiple keys on single node," J. IPS Japan, vol.49, no.9, pp.3223–3233, Sept. 2008 (in Japanese).
[11] R. Banno, T. Fujino, S. Takeuchi, and M. Takemoto, "SFB: A scalable method for handling range queries on Skip Graphs," IEICE Commun. Express, vol.4, no.1, pp.14–19, Jan. 2015.
[12] W. Pugh, "Skip Lists: A probabilistic alternative to balanced trees," Communications of the ACM, vol.33, no.6, pp.668–676, June 1990.
[13] A.G. Beltran, P. Sage, and P. Milligan, "Skip Tree Graph: A distributed and balanced search tree for peer-to-peer networks," Proc. IEEE International Conference on Communications, pp.1881–1886, June 2007.
[14] J. Vuillemin, "A data structure for manipulating priority queues," Communications of the ACM, vol.21, pp.309–315, April 1978.
[15] N.J.A. Harvey and J.I. Munro, "Deterministic SkipNet," Information Processing Letters, vol.90, no.4, pp.205–208, May 2004.
[16] T. Kawaguchi, R. Banno, M. Hojo, M. Ohnishi, and K. Shudo, "Self-Refining Skip Graph: Skip Graph approaching to an ideal topology," Proc. IEEE Annual Consumer Communications Networking Conference, pp.441–448, Jan. 2017.
[17] R. Banno, S. Takeuchi, M. Takemoto, T. Kawano, T. Kambayashi, and M. Matsuo, "Designing overlay networks for handling exhaust data in a distributed topic-based pub/sub architecture," Journal of Information Processing, vol.23, no.2, pp.105–116, March 2015.
[18] WonderNetwork, "Global Ping Statistics." https://wondernetwork.com/pings (accessed 2019-09-24).
[19] Google Developers, "Measure Performance with the RAIL Model." https://developers.google.com/web/fundamentals/performance/rail/ (accessed 2019-09-24).
[20] R. Kohavi and R. Longbotham, "Online Experiments: Lessons Learned," IEEE Computer, vol.40, no.9, pp.103–105, Sept. 2007.
[21] Medium, "India's Hotstar draws over 10 million concurrent viewers, sets new global record." https://medium.com/@refsrc/indias-streaming-service-hotstar-draws-over-10-million-concurrent-viewers-sets-new-global-record-29746440dfb4 (accessed 2019-09-24).

Ryohei Banno received his Bachelor's and Master's degrees from Hokkaido University in 2010 and 2012, then earned his Ph.D. from Tokyo Institute of Technology in 2018. He was a researcher in NTT Network Innovation Laboratories from 2012 to 2018. Since 2018, he has been a researcher in Tokyo Institute of Technology. He received IPSJ Best Paper Award in 2015. His research interests include distributed computing, publish/subscribe massaging, and IoT/M2M systems. He is a member of IEEE, IEEE Computer Society, IPSJ, and IEICE.

Kazuyuki Shudo received B.E. in 1996, M.E. in 1998, and a Ph.D. degree in 2001 all in computer science from Waseda University. He worked as a Research Associate at the same university from 1998 to 2001. He later served as a Research Scientist at National Institute of Advanced Industrial Science and Technology. In 2006, he joined Utagoe Inc. as a Director, Chief Technology Officer. Since December 2008, he currently serves as an Associate Professor at Tokyo Institute of Technology. His research interests include distributed computing, programming language systems and information security. He has received the best paper award at SACSIS 2006, Information Processing Society Japan (IPSJ) best paper award in 2006, the Super Creator certification by Japanese Ministry of Economy Trade and Industry (METI) and Information Technology Promotion Agency (IPA) in 2007, IPSJ Yamashita SIG Research Award in 2008, Funai Prize for Science in 2010, The Young Scientists' Prize, The Commendation for Science and Technology by the Minister of Education, Culture, Sports, and Technology in 2012, and IPSJ Nagao Special Researcher Award in 2013. He is a member of IEEE, IEEE Computer Society, IEEE Communications Society and ACM.