Adaptive Topology for Scalability and Immediacy in Distributed Publish/Subscribe Messaging

Ryohei Banno*†, Kazuyuki Shudo*
*Tokyo Institute of Technology, Tokyo, JAPAN
†Kogakuin University, Tokyo, JAPAN
Email: banno@computer.org

***Abstract*—Publish/subscribe is a communication model for exchanging messages via a broker while providing loose coupling. So far, several studies have been conducted to address load concentration on the broker by forming distributed brokers. However, although they achieve higher throughput by load distribution among multiple brokers, these existing studies require an increased latency for message delivery. In this paper, we propose a novel method to construct and maintain an adaptive topology that features both scalability and immediacy in distributed publish/subscribe messaging. The proposed method is for topic-based publish/subscribe systems and uses a number of brokers to form an overlay network. Its topology changes dynamically to compose a subgraph for each topic in a single-hop or multi-hop manner according to the topic load (i.e., the number of clients). The experimental results show that compared to existing studies, the proposed method reduces the delivery path length, which is a principal factor that affects latency. Especially for low load topics, the reduction rate of the proposed method reaches values greater than 60%.**

***Index Terms*—Overlay networks, Publish subscribe systems, Skip graph**

I. Introduction

Publish/subscribe is a communication model for exchanging messages via a server called a "broker", thus, it provides loose coupling [1]; the broker dynamically determines the data receiver based on the message content, so that the sender (publisher) and the receiver (subscriber) need not care about the communication target. Topic-based publish/subscribe is one of publish/subscribe messaging patterns and is widely used in various information systems. In topic-based publish/subscribe systems, publishers and subscribers exchange messages via logical channels called "topics" as shown in Figure 1. Subscribers register their interest in topics to the broker in advance and subsequently receive messages on their selected topics of interest.

Because topic-based publish/subscribe is suitable for unstable, i.e., frequently changed, relationships among low powered devices due to its loose coupling nature, it has attracted much academic and industrial interest for use in IoT systems. Indeed, its well-known protocol MQTT [2] has been introduced in various IoT platforms [3], [4], in addition to the OPC UA protocol, which is used in the reference architecture model for Industry 4.0 and supports topic-based publish/subscribe functionality [5].

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

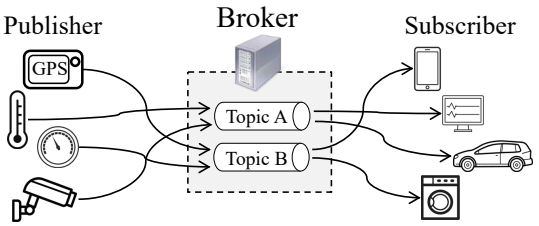

Fig. 1. Topic-based publish/subscribe

When considering a large-scale system composed of a vast number of IoT devices, the load concentration on the broker could cause service disruption or reduce the quality of service (QoS). To solve this problem, several existing studies for distributed publish/subscribe messaging have utilized multiple brokers [6]–[9]. In these studies, the brokers comprise an overlay network and work cooperatively to obtain a superior load distribution.

However, these approaches also increase the latency for message delivery despite achieving higher throughput by performing load distribution among multiple brokers. In our preliminary experiment, using Interworking Layer of Distributed MQTT brokers (ILDM) [10], [11], which enables multiple MQTT brokers to cooperate with each other, the relationship between throughput and latency is depicted in Figure 2. Note that in this experiment, we simulated the topology of SG-TBPS described later in Section II-A. The cooperation among the five brokers results in an approximately three times higher throughput compared to a single broker, whereas the latency increases to approximately six times higher. Although the experiment was done at a small scale and throughput and latency generally largely depend on the measurement environment and implementation quality, the results indicate that a trade-off exists to some extent between scalability and immediacy.

Because the result in Figure 2 was done in a LAN using only five brokers, cooperation among a larger number of brokers (e.g., tens of thousands) in an actual network environment is considered to be likely to involve latencies of dozens to hundreds of milliseconds. Such an increase in latency impairs

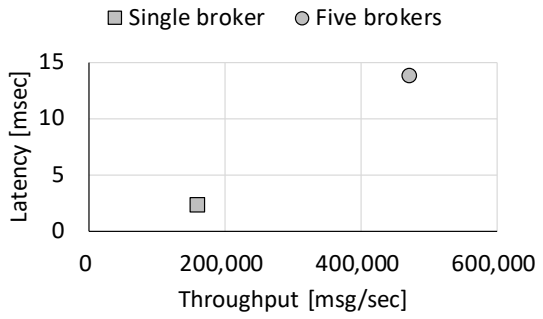

Fig. 2. Effect on throughput and latency by load distribution

the immediacy of publish/subscribe messaging and narrows its application range. For example, 100 milliseconds delay is said to substantially degrade the user experience (UX) for web contents [12] and reduces the product volumes sold in e-commerce systems [13]. In IoT systems especially those that require high immediacy, such as real-time device control, the latency requirements are expected to be more severe.

In this paper, we propose a novel method to construct and maintain an adaptive topology that features both scalability and immediacy in distributed topic-based publish/subscribe messaging. The proposed method focuses on the delivery path length based on the multi-hop forwarding schemes in existing studies, which is the primary factor that increases latency. Composing broker topology in a single-hop manner obviously reduces latency, but it also degrades throughput because multi-hop forwarding contributes to load distribution. To address this trade-off and obtain both scalability and immediacy, the topology of the proposed method is changed dynamically to compose a subgraph for each topic in a single-hop or multi-hop manner based on the topic load (i.e., the number of clients).

The remainder of this paper is organized as follows. Section II introduces related studies on distributed topic-based publish/subscribe messaging. Section III explains some elemental techniques, while Section IV describes the proposed method. Section V presents the experimental evaluation. Conclusions and suggestions for future work are provided in Section VI.

II. Related work

Several existing studies have utilized multiple brokers for topic-based publish/subscribe messaging.

Scribe [6] is a method for handling multicast groups using Pastry [14] which is one of the Distributed Hash Table (DHT) algorithms. In Scribe, a delivery tree is formed on the overlay network of Pastry for each multicast group. The node responsible for the group name key becomes the root node for the group. Reverse paths of routing from the group member nodes to the root node on the Pastry network compose the tree of the group. When a participant node sends a message to the root node, the message is forwarded along the tree and eventually delivered to all the group members. By creating a group for each topic, Scribe can be utilized for topic-based publish/subscribe messaging.

Bayeux [7] is similar to Scribe, but is based on Tapestry [15], another DHT algorithm that utilizes forward paths from the root node to form a delivery tree. Compared to Scribe, this method has a disadvantage that a root node must maintain information about all the group members.

CAN-MC [8] is also a multicast method based on a DHT algorithm called CAN [16]. Different from Scribe and Bayeux, CAN-MC uses two kinds of overlay networks: a global CAN network for finding a group and smaller CAN networks for disseminating a message within a group. Although CAN-MC has the advantage that a message is forwarded among only the corresponding group members, its overlay networks have relatively high maintenance costs.

A. SG-TBPS

Although the above existing studies achieve high scalability, they possess several inefficiencies. For example, for Scribe and Bayeux, the path length from a publisher to a subscriber is $O(\log N)$ where N is the number of nodes. Note that here and hereafter we call a broker connected to a publisher client of topic t "a publisher of topic t", as well as we call a broker connected to a subscriber client of topic t "a subscriber of topic t". Even when the number of subscribers of the corresponding topic is quite small (e.g., only one), each message is forwarded by $O(\log N)$ nodes; thus, the latency increases, and network resources are wasted. In addition, for either method, a published message will be wastefully forwarded even when no corresponding subscriber exists.

These problems become more serious when considering the "data exhaust" in IoT [17]. Namely, IoT data has low value density; it consists of a small amount of valuable data and a large amount of less-valuable data. Considering this IoT data, topics with no or few subscribers are dominant like power-law, and the above problems become prominent.

To overcome this, a method using Skip Graph [18] was proposed [9]. Hereafter, we call this method Skip Graph-based topic-based publish/subscribe (SG-TBPS). In SG-TBPS, a number of brokers are placed at the edge of a wide area network as shown in Figure 3, composing a Skip Graph topology. SG-TBPS lets publishers and subscribers of a topic form connected subgraphs, and the two subgraphs of a topic are also connected. Accordingly, the following features are obtained.

- The path length from a publisher to a subscriber is $O(\log M)$ where M is the number of nodes interested in the corresponding topic.
- Publishers can suspend publishing by detecting the absence of the corresponding subscribers.

We explain the topological details of SG-TBPS in detail later in Section III-C.

Although SG-TBPS is more suitable for IoT data compared to the other above mentioned methods, it still has a problem of increasing latency as shown in Figure 2.

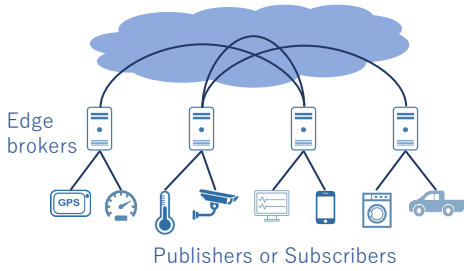

Fig. 3. Architecture of SG-TBPS

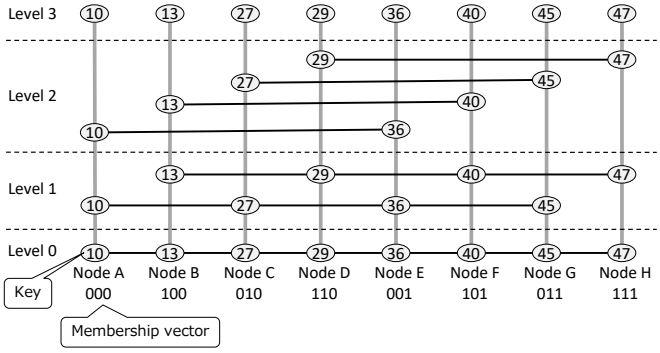

Fig. 4. Example of Skip Graph

III. Preliminaries

In this section, we explain some of the elemental techniques utilized in our proposed method.

A. Skip Graph

Skip Graph [18] is a type of structured overlay network known for its ability to handle range queries. Each node has a key and a random sequence called a membership vector. In this paper, we assume the alphabet $\Sigma = 0, 1$ for membership vectors.

As shown in Figure 4, a Skip Graph topology composes a multiplex structure of skip list [19]. It has a hierarchical structure; Level 0 is a doubly linked list containing all the nodes sorted by key, while in Level i, the nodes whose membership vectors have the same i-digit prefix compose a doubly linked list. Any node can search for another node by specifying a key. The node search begins from the maximum level of the start node, then the query is forwarded to the farthest node that does not exceed the target key and gradually moves down to Level 0. Thus, the path length is $O(\log N)$ for N nodes.

B. Range queries in Skip Graph

Since nodes in Skip Graph are sorted by keys, which differs from DHTs, any node can search for other nodes by specifying a key range.

SFB [20], [21] is one of the methods for routing range queries in Skip Graph. It introduces a divide-and conquer approach in which the target range is divided into subranges

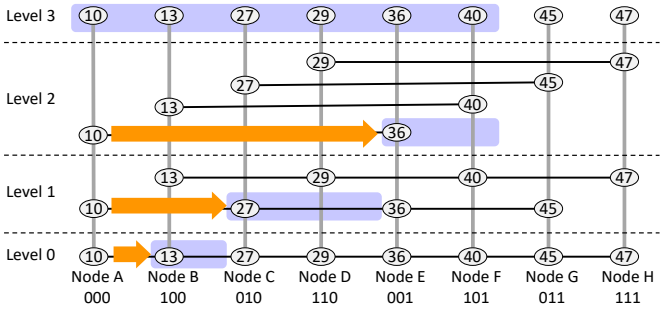

Fig. 5. Routing range query by SFB

and processed recursively. Figure 5 shows an example. If node A issues a range query with a target range from 10 to 40, the range is divided by the keys of the neighbors of node A. Each subrange is forwarded to the neighbors and processed in the same manner.

SFB has a shorter path length and a smaller number of messages than do other methods, such as MRF [22] or sequential traversing and broadcasting [23].

C. Topology of SG-TBPS

As described in Section II-A, SG-TBPS is a method for distributed toipc-based publish/subscribe messaging using Skip Graph. Brokers placed at the edge of a wide area network compose an overlay network. Each broker generates a key from its topic name of interest and its node types: publishers or subscribers. By using this key, the brokers maintain the virtual nodes of Skip Graph and form a topology, as depicted in Figure 6.

Thus, the publishers and subscribers of a topic form connected subgraphs respectively, and the two subgraphs of a topic are also connected. This scheme results in the existence of a unique publisher node for each topic, called a rendezvous point, which is placed adjacent to the subgraph of the subscribers of the same topic in Level 0. The rendezvous point can detect when the corresponding subscribers are absent simply by checking its right-hand neighbor for Level 0. When the rendezvous point detects the absence of subscribers, it notifies the other publishers of that topic so that they can suspend publishing.

D. Flexible Routing Tables

Flexible Routing Tables (FRT) [24] is a method for maintaining routing tables flexibly in structured overlay networks. Generally, in the algorithms of structured overlay networks such as DHTs [14], [15], [25] and Skip Graph, a node's routing table is constructed by adding node information (entries) that satisfy certain conditions defined in each algorithm. For example, in Skip Graph, each routing table entry satisfies the condition that the membership vectors have a corresponding common prefix. In contrast, FRT uses the following two steps to construct a routing table.

1) Entry learning: add entries without any conditions.

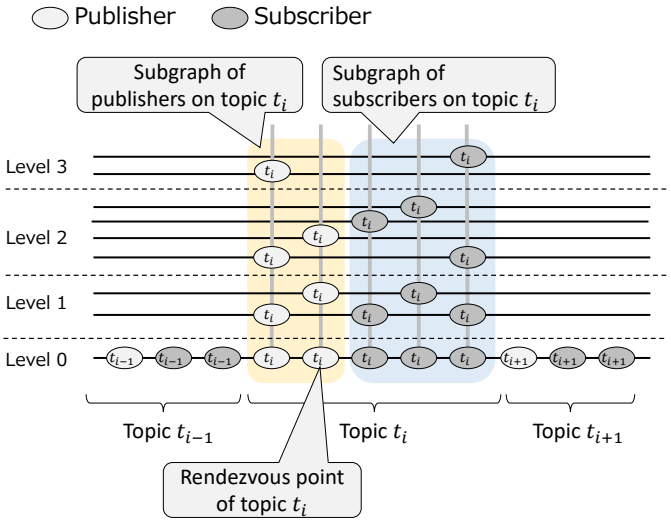

Fig. 6. Topology of SG-TBPS

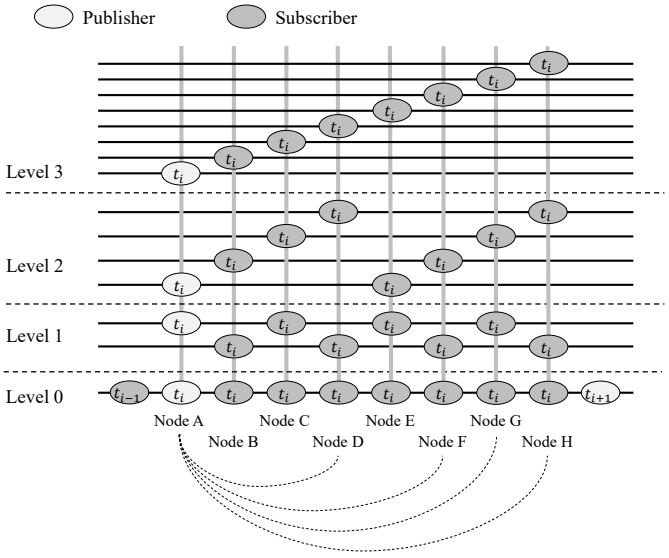

Fig. 7. Topology of ATM-TBPS

2) Entry filtering: remove one entry when the number of entries exceeds the given maximum size L.

Entry learning can be done by actively issuing a search query for a random target key or by passively using information obtained when forwarding messages from other nodes.

In entry filtering, the entry to be removed is selected from entries other than the mandatory entries called "sticky entries". The selection process is conducted by using an order relationship $\leq_{FRT}$ among routing tables, which determines the relative superiority or inferiority among routing tables; for example, $E \leq_{FRT} F$ means that routing table F is superior to routing table E. Among all the possible routing tables creatable by removing one entry from the current routing table, the best routing table is selected in accordance with $\leq_{FRT}$ and the corresponding entry will be removed. By defining $\leq_{FRT}$ appropriately according to each application, we can flexibly construct a variety of structured overlay networks.

One of the advantages of FRT is that it can construct both single-hop and multi-hop overlay networks using the same algorithm. That is, when the number of nodes is equal to or less than the maximum routing table size L, each node can have the entries of all other nodes in its routing table; thus the topology becomes single-hop. However, when the number of nodes exceeds L, the topology becomes multi-hop based on the entry filtering process. In contrast, in structured overlay networks without FRT, each node does not possess the entries of nodes that do not satisfy certain conditions in its routing table. Therefore, even when the number of nodes is small, the topology largely remains multi-hop.

IV. Proposed method

As described in Section I, the existing methods described in Section II involve an increase of latency in return for higher scalability. In this section, we explain the proposed method for achieving both scalability and immediacy. Hereafter, we call the proposed method the adaptive topology management for topic-based publish/subscribe (ATM-TBPS).

A. Order relationship among routing tables

ATM-TBPS introduces the concept of FRT into SG-TBPS. In ATM-TBPS, $\leq_{FRT}$ is defined so that the routing table that includes the subscriber nodes of the same topic becomes superior to those that do not include the subscriber nodes. This enables nodes related to a topic with a small number of subscribers to form a single-hop, i.e., a full mesh, subgraph. On the other hand, nodes related to a topic with a large number of subscribers form a multi-hop subgraph to avoid concentrating the message delivery load. Using this approach, we can obtain both scalability and immediacy: low latency for low-load topics and high throughput for high-load topics.

We define $\leq_{FRT}$ by extending FRT-Skip Graph [26] to keep the entries of subscribers of the same topic in routing tables with high priority. Specifically, the following policies are applied in turn to determine the superiority of routing tables. Note that entries corresponding to the neighbors for Level 0 in Skip Graph are sticky entries.

1) Keep entries corresponding to neighbors for each level in Skip Graph.
2) Keep entries of subscribers of the same topic.
3) Keep the same number of entries for each level.
4) Keep entries of nodes whose membership vectors have a longer common prefix.
5) Keep entries of nodes whose keys are closer.

Consider the example shown in Figure 7. Node A has preferential entries for nodes B, C, and E in its routing table because they correspond to its neighbors for each level in Skip Graph. When the maximum routing table size L of node A is sufficiently large, there is a possibility that entries for nodes D, F, G, and H will be added to node A's routing table. The

$$
\begin{aligned}
B_1(E,F) &\Leftrightarrow l_{max}(E) < l_{max}(F) \ \vee \ (l_{max}(E) = l_{max}(F) \ \wedge \ |EMP(E)| > |EMP(F)|) \\
B_2(E,F) &\Leftrightarrow l_{max}(E) = l_{max}(F) \ \wedge \ |EMP(E)| = |EMP(F)| \\
B_3(E,F) &\Leftrightarrow B_2 \ \wedge \ |SUB(E)| < |SUB(F)| \\
B_4(E,F) &\Leftrightarrow B_2 \ \wedge \ |SUB(E)| = |SUB(F)| \\
B_5(E,F) &\Leftrightarrow B_4 \ \wedge \ |MAX(E)| < |MAX(F)| \\
B_6(E,F) &\Leftrightarrow B_4 \ \wedge \ |MAX(E)| = |MAX(F)| \\
B_7(E,F) &\Leftrightarrow B_6 \ \wedge \ min(MAX(E)) < min(MAX(F)) \\
B_8(E,F) &\Leftrightarrow B_6 \ \wedge \ min(MAX(E)) = min(MAX(F)) \\
B_9(E,F) &\Leftrightarrow B_8 \ \wedge \ KEY(SUCC(E)) >_{dic} KEY(SUCC(F)) \\
B_{10}(E,F) &\Leftrightarrow B_8 \ \wedge \ KEY(PRED(E)) >_{dic} KEY(PRED(F))
\end{aligned}
\tag{1}
$$

Entry	Level
Node B	0
Node C	1
Node D	0
Node E	2
Node F	0

$\leq_{FRT}$

Entry	Level
Node B	0
Node C	1
Node E	2
Node F	0
Node G	1

$\leq_{FRT}$

Entry	Level
Node B	0
Node C	1
Node D	0
Node E	2
Node G	1

Fig. 8. Superiority of routing tables

TABLE I
LIST OF NOTATIONS

Notation	Description
$l_{max}(E)$	The maximum level in routing table E.
$EMP(E)$	A set of levels for which routing table E has no entries. That is, $EMP(E) = \{l_i \mid \|E_{l_i}\| = 0\}$.
$SUB(E)$	A set of entries of subscribers whose topic is the same as X in routing table E.
$MAX(E)$	A set of levels for which routing table E has the most entries.
$SUCC(E)$	A set of entries on the right hand side of X in routing table E.
$PRED(E)$	A set of entries on the left hand side of X in routing table E.
$KEY(E)$	A sequence of entries sorted by key in routing table E.

dotted lines in Figure 7 indicate these potential links from node A. If the max level of node A is three and $L = 8$, node A is able to add two entries in addition to the entries corresponding to its three neighbors on both sides in Skip Graph. If node A acquires the entries for nodes D, F, and G by the entry learning process, it must remove one of the three entries by the entry filtering process. There are three possible routing tables and their order by $\leq_{FRT}$ is as shown in Figure 8. To preserve the same number of entries for each level insofar as possible, the removal of node G is determined to be inferior because nodes D and F are both Level 0 entries. Additionally, to keep the entries for nodes whose keys are closer, the removal of node D is determined to be inferior to the removal of node F, because the key of node D is closer to that of node A. Therefore, the entry for node F is removed in this case. Note that if L were larger than 10, node A could hold information for all the subscriber nodes of topic t_i (i.e., the topic t_i subgraph becomes a single-hop topology).

To define $\leq_{FRT}$ to satisfy the above policies, we first define some notations for the routing table E of node X as shown in Table I. Using these notations, we can define the propositions in Equation (1). Herein, $>_{dic}$ indicates lexicographic order; then we have

$$\{a_i\} <_{dic} \{b_i\} \Leftrightarrow a_k < b_k \quad (k = min\{i \mid a_i \neq b_i\}).$$

From the above propositions, the order relationship between routing table E and F in ATM-TBPS is defined as follows:

$$
\begin{aligned}
E \leq_{FRT} F \quad \Leftrightarrow \quad & B_1(E,F) \ \vee \ B_3(E,F) \\
& \vee \ B_5(E,F) \ \vee \ B_7(E,F) \\
& \vee \ B_9(E,F) \ \vee \ B_{10}(E,F)
\end{aligned}
$$

Reconsidering the example in Figure 8, in the case of focusing on the part depicted in Figure 7, we can see that the order of those three routing tables is following the above definitions.

Let us say we have the three routing tables $R_1 = \{B, C, D, E, F\}$, $R_2 = \{B, C, E, F, G\}$, and $R_3 = \{B, C, D, E, G\}$. For R_1 and R_2, $B_2(R_1, R_2)$ is true because

$$l_{max}(R_1) = l_{max}(R_2) = 3$$

and

$$|EMP(R_1)| = |EMP(R_2)| = 1.$$

$B_4(R_1, R_2)$ is also true because

$$|SUB(R_1)| = |SUB(R_2)| = 5.$$

Since

$$|MAX(R_1)| = 1 < |MAX(R_2)| = 2,$$

$B_5(R_1, R_2)$ is true and consequently we can find $R_1 \leq_{FRT} R_2$.

For R_2 and R_3, in the same manner, we can find that $B_2(R_2, R_3)$, $B_4(R_2, R_3)$, $B_6(R_2, R_3)$, and $B_8(R_2, R_3)$ are true because

$$
\begin{aligned}
l_{max}(R_2) = l_{max}(R_3) &= 3, \\
|EMP(R_2)| = |EMP(R_3)| &= 1, \\
|SUB(R_2)| = |SUB(R_3)| &= 5, \\
|MAX(R_2)| = |MAX(R_3)| &= 2, \\
min(MAX(R_2)) = min(MAX(R_3)) &= 0.
\end{aligned}
$$

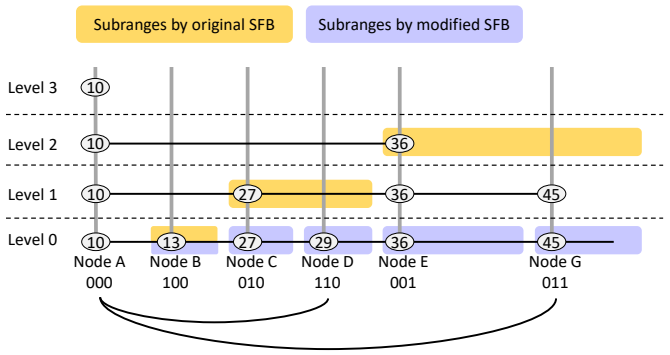

Fig. 9. Difference in dividing target range between SFB and modified SFB

$B_9(R_2, R_3)$ is also true because

$$KEY(SUCC(R2)) >_{dic} KEY(SUCC(R3)),$$

and consequently we can find $R_2 \leq_{FRT} R_3$.

B. Queries in ATM-TBPS

In ATM-TBPS, the queries listed in Table II are used to maintain the routing tables. Note that both the sender and forwarder information contain the IP address, the membership vector, and the key of the corresponding node.

The following sections explain the process that occurs each node when using these queries[1]. Hereafter, we notate the node corresponding to the neighbor for Level i in Skip Graph as "Level i neighbor node".

C. Routing messages

For exchanging messages and management notification among nodes, ATM-TBPS utilizes SFB described in Section III-B.

Note that we slightly modify the algorithm of SFB for ATM-TBPS, because it assumes a normal Skip Graph topology. In SFB, the key used to divide a target range is chosen in the order from the top-most level within the range. However, in the topology of ATM-TBPS, a routing table of a node could have multiple entries for the same level, unlike Skip Graph. That is, an entry for a level l_i could be farther than another entry for a level l_j even if $l_i < l_j$. For this reason, in ATM-TBPS, the key used to divide a target range is chosen in the order from the farthest key within the range.

Figure 9 shows an example difference in dividing the target range between SFB and the modified SFB in the case of Figure 8, i.e., Node A has entries of Node B, C, D, E, and G in its routing table. The target range is divided from the farthest key by using the modified SFB whereas it is divided from the top-most level neighboring key by using the original SFB.

[1]Note that the processes that are the same as SG-TBPS are omitted from this explanation.

D. Node join

When a new node joins a topic as a subscriber, it first searches its position by using a normal Skip Graph search query and adds both sides neighbors for Level 0 to its routing table as sticky entries. Then, it sends an REXP query with the search level set to 0 to the Level 0 neighbor nodes on both sides. This query is intended to collect the entries corresponding to neighbors for each level of Skip Graph.

After a certain period of time, if the number of entries is less than the maximum size L, the node issues an EEXP query targeting the range of subscribers of the same topic. If the number of entries is still less than L after a certain period of time, the node repeatedly issues an EEXP query to a random target.

E. Node leave

When a subscriber node leaves a topic, the node issues an UNSUB query targeted to the range of publishers of the topic. This query is intended to allow the removal of the entry of the leaving node from the routing tables of the publishers.

Basically, nodes do not manage information regarding which nodes have an entry for their own information, because the routing table construction is asymmetrical in FRT. However, removing disabled entries by using the timeout technique greatly impairs immediacy. Therefore, a departing node issues an UNSUB query as described above.

F. Publish

A publisher node can deliver a PUB query to all the corresponding subscribers by using SFB as described above. If the number of entries is less than the maximum size L, an EEXP query is piggy-backed onto the PUB query, enabling rapid identification of the subscriber nodes for the given topic.

G. Normal process

Each node sends a REXP query with the search level set to 0 to its Level 0 neighbor nodes on both sides at regular intervals. This query is intended to allow entries to be updated corresponding to the neighbors for each level of Skip Graph.

H. Receiving query

When a node receives a REXP query, it sends a NEW query to the sender node if the list of existing entries in the REXP query does not include itself. If the search level of the REXP query is i and the node corresponds to the Level $i+1$ neighbor node of the sender node, the node increases the search level by 1 and forwards the query to its Level $i+1$ neighbor node. If the node does not correspond to the Level $i+1$ neighbor node of the sender node, the node simply forwards the query to its Level i neighbor node. When no corresponding node exists to which to forward to the query, the node terminates the process.

When a node receives an EEXP query, it sends a NEW query to the sender node if the list of existing entries in the EEXP query does not include itself. The node decreases the number of wanted entries in the query by 1; then, it forwards

TABLE II
QUERIES FOR MAINTAINING ROUTING TABLES

Query	Description	Contained information
REXP	Query for regular exploration of entries	• Sender information (IP address, membership vector, key, etc.) • Forwarder information (IP address, membership vector, key, etc.) • List of existing entries • Search level
EEXP	Query for extra exploration of entries	• Sender information (IP address, membership vector, key, etc.) • Forwarder information (IP address, membership vector, key, etc.) • List of existing entries • Number of wanted entries
PUB	Query for publishing a message	• Sender information (IP address, membership vector, key, etc.) • Topic • Published message • Forwarder information (IP address, membership vector, key, etc.)
UNSUB	Query for notifying unsubscribing from a topic	• Sender information (IP address, membership vector, key, etc.) • Forwarder information (IP address, membership vector, key, etc.)
NEW	Query for notifying a new candidate of entry	• Sender information (IP address, membership vector, key, etc.)

the query to the corresponding destination if the number of wanted entries is still greater than 1.

When a node receives an UNSUB query, it removes the entry of the sender node from its routing table.

For every query received by a node, the node adds the sender node and the forwarder node to its routing table if they are not already present; however, the sender node of an UNSUB query is not added.

V. EVALUATION

We evaluated ATM-TBPS through simulation experiments. The simulation program was implemented in Java.

The number of brokers was $10,000$. Each broker joined 5 topics on average as either a publisher or a subscriber. We used one topic for measurement; the topic had one publisher and a variable number of subscribers ranging from 10 to $5,120$. We measured the average and maximum path lengths. Each measurement was conducted three times for each setting, and the result was calculated as the average value of those three measurements.

We used Scribe and SG-TBPS, described in Section II, as comparison targets. The value of L, the maximum size of a routing table, was set to 60 or 120. Note that the maximum size of routing tables in SG-TBPS was 58. Unlike ATM-TBPS, the size of a routing table is decided probabilistically in SG-TBPS and Scribe. This means that each node must be capable of the largest possible routing table size in the existing methods.

Figure 10 shows the result of the average path length. For ATM-TBPS, the path length becomes 1 when the number of subscribers is small. When executing ATM-TBPS with L set to 120, single-hop delivery was achieved for up to 80 subscribers. Compared to SG-TBPS, the reduction rate of the average path length of ATM-TBPS with $L = 60$ is more than 60% for up to 80 subscribers and more than 20% even for larger numbers of subscribers.

Figure 11 shows the maximum path length. For SG-TBPS and Scribe, the maximum path length is larger than the average path length. For ATM-TBPS, the maximum path length is 1 for small numbers of subscribers.

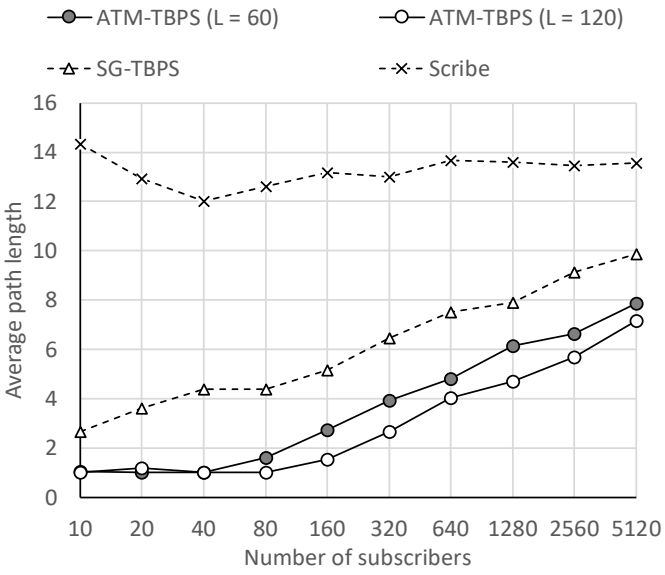

Fig. 10. Average path length

A. Discussion on latency

We compared latency using simple modeling. We assumed the following delay time:

- The communication delay between nodes is 10 milliseconds.
- The processing delay of a node on the message delivery path is 1 millisecond.
- The processing delay of a node for its each child node on the message delivery path is 0.1 millisecond.

Figure 12 shows the results. Note that we calculated latency for a single broker case besides ATM-TBPS, SG-TBPS, and Scribe. It was approximated by considering the case that the number of child nodes was equal to the number of subscribers.

For Scribe and SG-TBPS, the latency becomes worse than a single broker when the number of subscribers is relatively small, although they achieve better latency with larger numbers

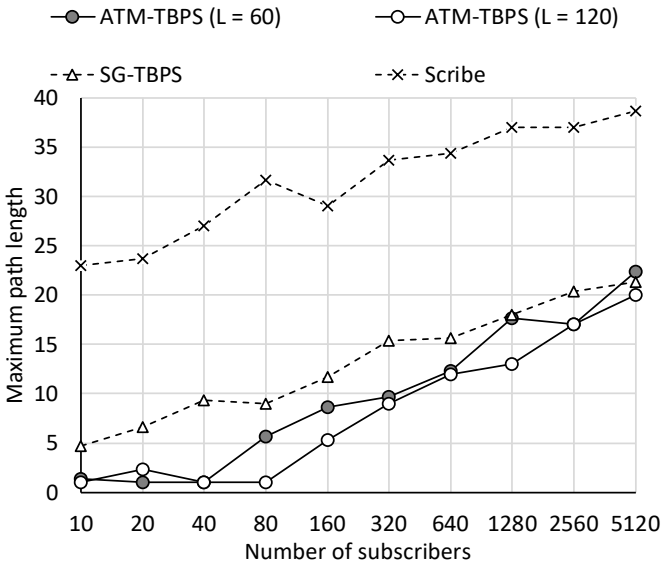

Fig. 11. Maximum path length

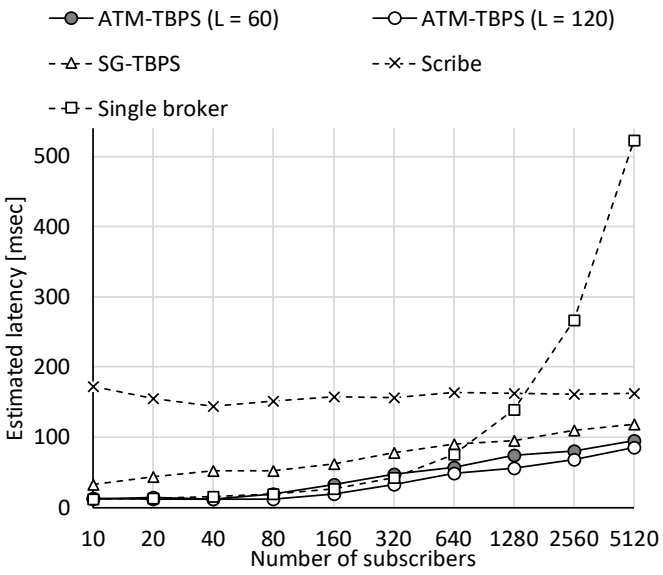

Fig. 12. Estimated latency

of subscribers. However, this inefficiency for small numbers of subscribers is not desirable for IoT systems, because topics with no or few subscribers are considered to be dominant, as described in Section I. In contrast, using ATM-TBPS, it can be seen that both scalability and immediacy are obtained; the latency is comparable to that of a single broker for small numbers of subscribers, and remains comparable to the latencies of previous methods for larger numbers of subscribers.

One thing to be mentioned is throughput performance of ATM-TBPS. Throughput is affected by various factors such as bandwidth and RTT among nodes, underlying protocols, and occurrence of re-transmission. Assuming a single topic and a homogeneous network environment, the dissemination load of each node is considered to be proportional to the number of child nodes, i.e., neighbor nodes to be forwarded a message. In other words, throughput performance is suppressed to the value dividing the bandwidth with the number of child nodes. ATM-TBPS can keep the number of child nodes small by the entry filtering process even if the number of subscribers for the topic is quite large. This enables ATM-TBPS to have higher throughput than a single broker. Although quantitative evaluation of throughput performance is out of the scope of this paper, it is one of our future works.

VI. Conclusion

In this paper, we proposed a novel method called ATM-TBPS that achieves both scalability and immediacy for distributed topic-based publish/subscribe messaging. By utilizing the FRT concept, the topology of ATM-TBPS changes dynamically to compose a subgraph for each topic in either a single or multi-hop manner based on the topic load (i.e., the number of subscribers). The experimental results show that ATM-TBPS reduces the delivery path length and latency compared to existing studies, especially for low-load topics.

In future work, we plan to conduct an evaluation of throughput performance with actual protocols e.g., MQTT and more practical experiments such as simulating nodes joining or leaving.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

[2] Akamai IoT Edge Connect, https://developer.akamai.com/iot-edge-connect/ (accessed 2020-01-30).

[3] Amazon Web Services, "Designing MQTT Topics for AWS IoT Core," May 2019.

[4] MQTT, https://mqtt.org/ (accessed 2020-01-30).

[5] OPC Foundation, "OPC Foundation announces OPC UA PubSub release as important extension of OPC UA communication platform," https://opcfoundation.org/news/press-releases/opc-foundation-announces-opc-ua-pubsub-release-important-extension-opc-ua-communication-platform/ (accessed 2020-01-30), March 2018.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications*, vol. 20, no. 8, pp. 1489–1499, December 2002.

[7] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux : An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination," in *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video*, June 2001, pp. 11–20.

[8] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast using Content-Addressable Networks," in *Proc. International COST264 Workshop on Networked Group Communication*, November 2001, pp. 14–29.

[9] 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, 2015.

[10] R. Banno, J. Sun, M. Fujita, S. Takeuchi, and K. Shudo, "Dissemination of edge-heavy data on heterogeneous MQTT brokers," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, September 2017, pp. 1–7.

[11] R. Banno, J. Sun, S. Takeuchi, and K. Shudo, "Interworking Layer of Distributed MQTT Brokers," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 12, pp. 2281–2294, 2019.

[12] Google, "Measure Performance with the RAIL Model," https://developers.google.com/web/fundamentals/performance/rail/ (accessed 2020-01-30).
[13] R. Kohavi and R. Longbotham, "Online Experiments: Lessons Learned," *IEEE Computer*, vol. 40, no. 9, pp. 103–105, September 2007.
[14] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, November 2001, pp. 329–350.
[15] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, January 2004.
[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable Content-Addressable Network," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 161–172, October 2001.
[17] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, *Big data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute, 2011.
[18] J. Aspnes and G. Shah, "Skip Graphs," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, pp. 37:1–37:25, November 2007.
[19] W. Pugh, "Skip Lists : A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, June 1990.
[20] R. Banno, T. Fujino, S. Takeuchi, and M. Takemoto, "SFB: A Scalable Method for Handling Range Queries on Skip Graphs," *IEICE Communications Express*, vol. 4, no. 1, pp. 14–19, January 2015.
[21] R. Banno and K. Shudo, "An Efficient Routing Method for Range Queries in Skip Graph," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 3, 2020.
[22] 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," *Journal of Information Processing Society of Japan* (in Japanese), vol. 49, no. 9, pp. 3223–3233, September 2008.
[23] A. G. Beltran, P. Milligan, and P. Sage, "Range queries over skip tree graphs," *Computer Communications*, vol. 31, no. 2, pp. 358–374, February 2008.
[24] 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 International Conference on Peer-to-Peer Computing*, August 2011, pp. 72–81.
[25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, October 2001.
[26] M. Hojo, R. Banno, and K. Shudo, "FRT-Skip Graph: A Skip Graph-style structured overlay based on Flexible Routing Tables," in *Proc. IEEE Symposium on Computers and Communication (ISCC)*, June 2016, pp. 657–662.