Journal of Information Processing Vol.23 No.2 105–116 (Mar. 2015)

[DOI: 10.2197/ipsjjip.23.105]

Regular Paper

Designing Overlay Networks for Handling Exhaust Data in a Distributed Topic-based Pub/Sub Architecture

Ryohei Banno[1,a)] Susumu Takeuchi[1] Michiharu Takemoto[1] Tetsuo Kawano[1] Takashi Kambayashi[2] Masato Matsuo[1]

Received: May 12, 2014, Accepted: September 12, 2014

Abstract: To provide event-driven services in IoT, scalable methods of topic-based pub/sub messaging are indispensable. Methods using structured overlay networks are promising candidates. However, existing methods have the problem of wasting network resources, because they lack adaptivity to "exhaust data," which have low or no value most of the time. The problem contains two aspects. One is that each publisher node continues to forward data to a relay node even if there are no subscribers. The other is that excessively large multicast trees are constructed for low value data, which will be received by only a small number of subscribers. In this paper, we formulate the desirable design of overlay networks by defining a property called "strong relay-free" as an expansion of relay-free property. The property involves publishers and subscribers composing connected subgraphs to enable detecting the absence of subscribers and autonomously adjusting the tree size. We also propose a practical method satisfying the property by using Skip Graph, and evaluate it through simulation experiments. We confirmed that the proposed method can suspend publishing adaptively, and shorten the path length on multicast trees by more than 75% under an experimental condition with 100,000 nodes.

Keywords: distributed pub/sub, overlay networks, Skip Graph, relay-free, exhaust data, IoT

1. Introduction

The number of Internet-connected devices has been increasing and is estimated to reach 100 billion by 2020 [1]. This indicates the coming of the Internet of Things (IoT) and will bring about various smart services that are typically event-driven, i.e., controlling devices in accordance with some kind of event in the real space observed by sensors.

To deliver sensor data in real-time efficiently, pub/sub messaging [2] is required instead of using traditional request-reply messaging. In topic-based pub/sub, messages are exchanged through logical channels called "topics." Users subscribe to topics of interest and receive messages published on those topics. This paradigm provides convenient decoupling between publishers and subscribers, e.g., each publisher has no concern with the location of subscribers that will receive a published message.

Since IoT is considered to consist of a vast number of devices which generate massive stream data, the architecture and algorithm of pub/sub messaging should have high scalability.

In this paper, we focus on structured overlay networks known primarily for Distributed Hash Tables (DHTs) [3], [4], [5]. They have suitable properties such as scalability, robustness, and elimination of a single point of failure. Methods of topic-based pub/sub using DHTs have been proposed [6], [7], [8].

However, these methods do not work efficiently for a certain kind of data called "exhaust data," which is predicted to occupy most of the IoT data [9]. Exhaust data have low value density, namely the data have low or no value most of the time. Because the existing methods are not adaptive to the transition of the value of data, they waste network resources by gratuitously ventilating low/no valued data.

For overcoming such inefficiencies, we first clarify the requirements of overlay networks for improving the adaptivity and define a desirable property called "strong relay-free" by expanding relay-free, which is mainly known in studies based on unstructured overlay networks [10], [11]. In the strong relay-free property, publishers and subscribers compose connected subgraphs respectively, and these subgraphs are also connected to each other. This allows detecting whether there are subscribers or not, and autonomously adjusting the forwarding path lengths by the number of subscribers and publishers.

Subsequently, we propose a method for constructing overlay networks satisfying the property using Skip Graph [12]. To evaluate the proposed method, we implemented simulated programs for it and one of the DHT-based methods, and conducted some experiments with up to approximately 100,000 nodes. We confirmed that the proposed method can reduce consumption of network resources by suspending publishing adaptively, and can shorten the path length compared to the existing method. The experimental results also indicate that our method can predict the load on each node unlike with the conventional method.

Hence, the contributions of this paper are threefold:

- First, we give an architecture called "edge broker model"

[1] NTT Network Innovation Laboratories, NTT Corporation, Musashino, Tokyo 180–8585, Japan
[2] NTT Science and Core Technology Laboratory Group, NTT Corporation, Atsugi, Kanagawa 243–0198, Japan
[a)] banno.ryohei@lab.ntt.co.jp

which is suitable for handling exhaust data.
- Second, we present a novel design of overlay networks by defining "strong relay-free" property.
- Third, we provide a practical method satisfying the property by using Skip Graph.

The rest of this paper is organized as follows. Section 2 illustrates an architecture suitable for exhaust data streams and explains technical problems. Section 3 introduces related studies on topic-based pub/sub methods based on structured overlay networks, with a discussion of their inadequacies. Section 4 clarifies the requirements for constructing desirable overlay networks and formulates them as the strong relay-free property, while a practical method satisfying the property using Skip Graph is described in Section 5. Section 6 shows the results of simulation experiments to confirm the effectiveness of the proposed method. Finally, we summarize and conclude this paper in Section 7.

2. Edge Broker-based Architecture

Conventional topic-based pub/sub architectures have a broker server for managing topics [13], [14]. The broker gathers all published messages and processes filtering and forwarding to corresponding subscribers. All published messages are sent to the broker, then they are filtered and forwarded to corresponding subscribers. In other words, these architectures form a centralized model, which is easy to implement and commonly used for a wide variety of applications such as RSS, distribution of disaster prevention information, SNS, video chat, and so on.

However, the above centralized model does not work efficiently for exhaust data. The characteristics of exhaust data can be described as follows:
- Low value density
  Data are generated as byproducts and without specific uses. These data have low or no value most of the time, but sometimes are highly useful such as drive recorders.
- High frequency
  Data are automatically and continuously generated by devices, unlike today's Internet in which humans generate most of the content.
- Wide area
  Data are generated over a wide area in the physical space.

Namely, a tremendous amount of published data is concentrated on the broker with oppressing the network bandwidth. This is unprofitable because the data arriving at the broker are mostly discarded due to its low value density (see **Fig. 1** (a)).

Accordingly, we suppose a model using edge brokers as shown in Fig. 1 (b). From the view point of each device, edge brokers are placed in front of the Internet depicted as a cloud in the figure. It means the brokers are installed over a wide area. Published data and subscriptions are collected at the closest one. If the edge brokers could exchange only essential data, this architecture prevents imprudent forwarding of exhaust data to the Internet. Such concepts of focusing on the periphery of the outer edge of core networks has become increasingly important, e.g., the proposal of "Edge computing" [15].

We now give an example to discuss the requirements of the edge broker model. A video stream captured with a camera sensor in a city is often useless because it captures just ordinary unexciting events. However, sometimes it can be used, for example, monitoring students on their way to school. If there is no subscriber or there are only subscribers joining the same edge broker, the video stream should not waste the resources of the core network. On the other hand, when events, e.g., flash mobs, occur, it may be required to forward the video stream to devices joining other edge brokers through the Internet. Possibly these events lead to a flash crowd, which needs scalable multicast mechanisms.

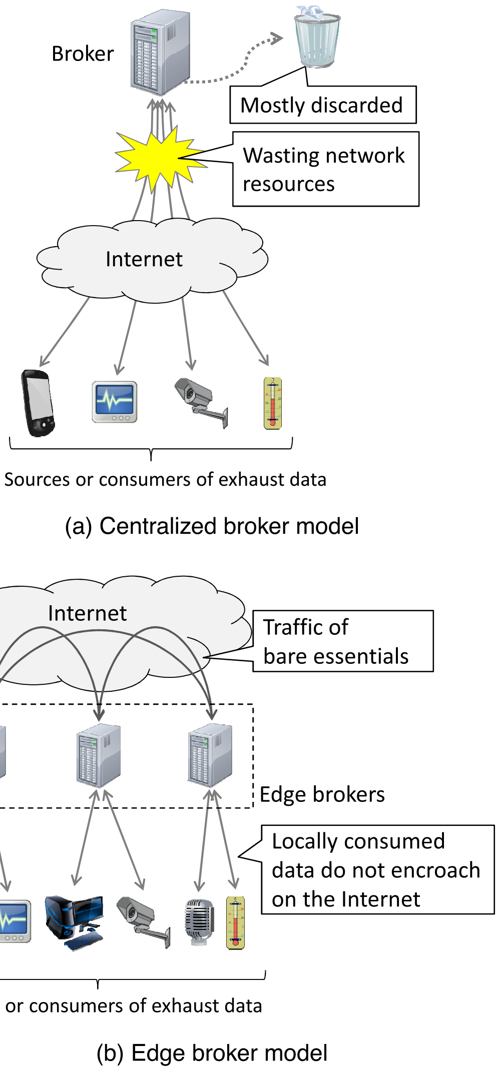

Broker
Mostly discarded
Wasting network resources
Internet
Sources or consumers of exhaust data
(a) Centralized broker model
Internet
Traffic of bare essentials
Edge brokers
Locally consumed data do not encroach on the Internet
Sources or consumers of exhaust data
(b) Edge broker model

Fig. 1 Pub/sub architectures for exhaust data streams.

Our aim with this research is to enable pub/sub messaging between edge brokers considering the above issues. An edge broker becomes a subscriber/publisher of a topic if one of the joining devices attempts to subscribe/publish to the topic. Hereafter, a subscriber or publisher means an edge broker playing the role of the subscriber or publisher, except when we explicitly mention the joining device.

3. Related Work

In this section, we provide an explanation of current methods of topic-based pub/sub messaging using structured overlay networks. These methods use DHTs in common and have been

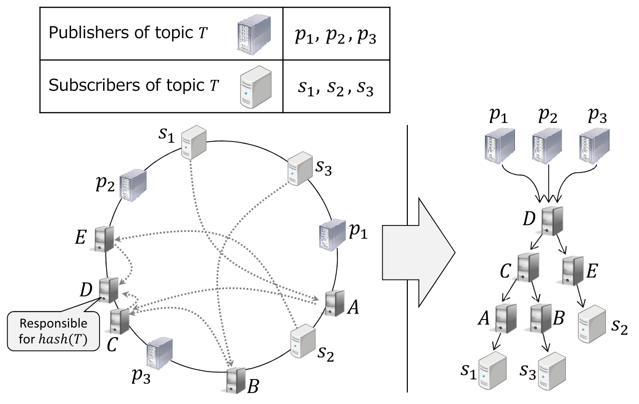


Fig. 2 Topic-based pub/sub by Scribe.

proposed as application layer multicast (ALM), where multicast groups correspond to topics in topic-based pub/sub.

3.1 Topic-based Pub/sub Method Using DHTs

Scribe [6] is an algorithm for achieving topic-based pub/sub and uses the Pastry network [3]. In Scribe, nodes form a tree for every topic. Each topic has a unique ID computed from a topic name by using a hash function, e.g., SHA-256. A node responsible for the ID on the Pastry network becomes the root node of the tree of that topic. The root node is called the rendezvous point while the other nodes of the tree are called forwarders. A node that attempts to subscribe to a topic sends a JOIN message towards the rendezvous point, according to the Pastry's routing protocol. A node that receives the message adds the joining node to its children table. If it had not been a forwarder of the topic before receiving the message, it forwards the JOIN message towards the rendezvous point. Therefore, the multicast path from the rendezvous point to the joining node is constructed in the reverse order of the Pastry's routing path as shown in **Fig. 2**. Publishers of the topic send messages towards the rendezvous point whose address can be found by the Pastry's routing protocol. Publishers can cache the address and send messages to the rendezvous point directly. Published messages are forwarded along the tree and delivered to all the corresponding subscribers.

Bayeux [7] is built on top of Tapestry [5] and achieves topic-based pub/sub in a similar way to Scribe. The primary difference is that Bayeux uses the forward-path forwarding scheme, while Scribe uses the reverse-path forwarding scheme [16]. In Bayeux, a node attempting to subscribe to a topic sends a JOIN message towards the root node, and each intermediate node in the path from the joining node to the root node simply forwards the message. When the root node receives the message, it sends a TREE message towards the joining node. Each node in the path from the root node to the joining node registers the joining node in its table.

CAN-MC [8], built on top of CAN [4], has presented a somewhat different style compared to the above two methods. CAN-MC consists of two types of CAN networks: the entire CAN and the mini CAN. The entire CAN is joined by all of the nodes and provides the function of looking up an introducer node, which is specific for each topic. The mini CAN is constructed for each topic independently and joined by the nodes of the topic. A published message is delivered by flooding over the corresponding mini CAN as follows:

- A publisher sends a message to all its neighbors.
- Each node receiving the message from its neighbor along dimension i forwards it to nodes as follows: the neighbors along dimension 1 to $(i-1)$, and those along dimension i in the opposite of the receiving direction.
- Each node does not forward the message along a particular dimension if it has already traversed at least half-way across the space from the publisher's coordinates along the dimension.
- Each node caches the sequence number of messages and does not process a duplicated message.

DYNATOPS [17] is an approach to extend rendezvous-based pub/sub methods like Scribe or Bayeux by using following two dynamic mapping algorithms.

- Similarity-based user placement
- Broker network reconfiguration

The former is an algorithm which aims to map users with similar subscriptions to nearby brokers. It is useful for reducing the subscription management overhead at brokers. This algorithm can also be applied to our approach, but it is out of the scope of this paper. The latter is an algorithm for dynamic reconfiguration of overlay networks of brokers, which aims to reduce the unrelated relay brokers on topic routing trees. The goal of this algorithm is somewhat similar to our research which succeeds in eliminating unrelated relay brokers. One of the major differences is that DYNATOPS drives the reconfiguration process with overhead subsequent to the establishment of topic routing trees.

3.2 Inadequacy for Exhaust Data Streams

In the edge broker model, it is preferable that the pub/sub messaging works efficiently especially when the number of subscribers is small or zero, because exhaust data have low value densities as described in Section 2. However, conventional methods have the following inefficiencies for handling exhaust data streams on the edge broker model, though they achieve high scalability.

3.2.1 Inability to Suspend Publishing

Conventional methods cannot suspend publishing even if there are no subscribers. In Scribe and Bayeux, publishers have no way to detect the absence of subscribers, and have to continue to constantly send messages towards the root node of the corresponding topic. Even in DYNATOPS, there is the same problem. In CAN-MC, nodes join the mini CAN of the topic of interest without any distinctions between subscribers and publishers. As a result, each publisher is forced to continue to flood messages as long as there are other publishers.

3.2.2 Gratuitous Forwarding

Scribe and Bayeux construct a multicast tree for each topic. The path length from the root node to each subscriber is the same as the lookup path length of Pastry and Tapestry, namely $O(\log N)$ where N is the number of entire nodes. Because this length does not depend on the number of nodes that are of the topic, published messages are forced to be gratuitously forwarded along the excessively long path even if there are only few subscribers.

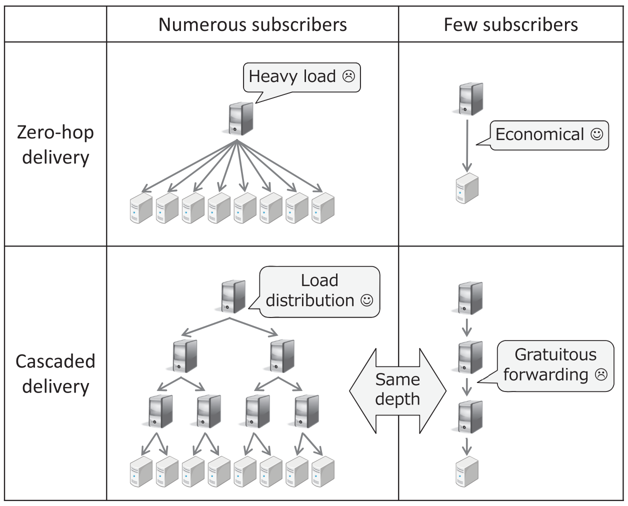


Fig. 3 Difference in forwarding costs and loads by number of subscribers.

This wastes network resources and increases the delay time of delivery.

Figure 3 illustrates this problem. As a primitive consideration, a heavy load is applied to the root node if it undertakes forwarding messages to all corresponding subscribers, as shown in the upper left of the figure. Scribe and Bayeux construct multicast trees to avoid this heavy load, as shown in the lower left of the figure. However, when there are only few subscribers, these methods force messages to be forwarded along the trees that have the same depth as the case of numerous subscribers (see the lower right of the figure). Regarding the case of few subscribers, the zero-hop delivery described in the upper right of the figure can forward more economically. It is also thought that most of the topics usually have few subscribers because of the low value density of exhaust data.

Thus, an efficient method is preferred, which achieves both economical forwarding for few subscribers and load distribution for numerous subscribers.

4. Formulation of Requirements

We first clarify the requirements for overcoming the problems described in Section 3.2.

- To prevent the inability to suspend publishing, it is required that all publishers should detect the switching between subscribers' absence and presence.
- To prevent gratuitous forwarding, it is required to shorten the path length for a small number of subscribers, while maintaining efficient load distribution of dissemination for a large number of subscribers.

Each requirement can be met in simple ways, such as broadcasting queries to determine the presence of subscribers, switching delivery mechanisms as shown in Fig. 3 for each topic, and so on. However these ways lack the global perspective and lead to other inefficiencies, e.g., negative effect on scalability. It is thus important for overlay networks to be constructed along a suitable design, which is a constraint in a sense. In the rest of this section, we focus on the "relay-free" property as an effective design of overlay networks.

4.1 Relay-free Property

The property of relay-free [10], also called Topic-connected Overlay, is primarily discussed in studies based on unstructured overlay networks [11]. The definition is as follows:

> Given a set of nodes V and a set of topics T, we define a Boolean-valued function $Int(v,t)$ with a node $v \in V$ and topic $t \in T$ as input. A node v is interested in a topic t if $Int(v,t) = true$, i.e., node v is a publisher or subscriber of topic t. Given an overlay network $G = (W, E)$, where $W = V$ and $E \subseteq V \times V$, G is relay-free if a subgraph of G induced by nodes $\{v \in V | Int(v,t) = true\}$ is connected for all $t \in T$.

In overlay networks that satisfy the relay-free property, a published message is forwarded only between nodes that are interested in the corresponding topic. It is expected that this property can contribute to shortening the path length for topics that have a small number of subscribers, because the diameter of the subgraph corresponding to each topic should become shorter in response to the decrease in subscribers. In the field of structured overlay networks, CAN-MC satisfies this property.

Satisfying the relay-free property provides suitability for shortening path length, but it is still difficult to suspend publishing when there are no subscribers. This is due to the fact that each publisher can obtain information on only its neighbors and there is no node having all information of the overlay network. Accordingly, a publisher cannot determine whether there is any subscriber not included in its neighbors.

4.2 Definition of Strong Relay-free Property

We newly define a desirable property called "strong relay-free" as an expansion of relay-free for suspending publications. In the strong relay-free property, we introduce the distinction between subscribers and publishers, unlike that these are treated as equivalent in relay-free. The definition is as follows, where V, T, G have the same meanings as above.

> We define a Boolean-valued function $Sub(v,t)$ and $Pub(v,t)$ with a node $v \in V$ and a topic $t \in T$ as input. A node v is a subscriber of a topic t if $Sub(v,t) = true$, and it is a publisher if $Pub(v,t) = true$. G is strong relay-free if all the following three conditions are satisfied for all $t \in T$:
>
> - A subgraph G_S induced by nodes $\{v \in V | Sub(v,t) = true\}$ is connected.
> - A subgraph G_P induced by nodes $\{v \in V | Pub(v,t) = true\}$ is connected.
> - A subgraph induced by G_S and G_P is connected.

In overlay networks satisfying the strong relay-free property, subscribers of each topic compose a connected subgraph. This means that the presence or absence of subscribers is synonymous with that of one subgraph. This is suitable for detecting the absence of subscribers under the constraint that each publisher has information only on its neighbors. Specifically, a publisher at the connection boundary between G_S and G_P seems possible to conclude whether subscribers are absent by checking only its neighbors. Furthermore, publishers of each topic also compose a connected subgraph, so one can easily disseminate the information about the absence of subscribers to others.

5. Efficient Topic-based Pub/sub Method

As mentioned previously, we have clarified the requirements and formulated them as a definition of the strong relay-free property. In this section, we propose a practical method for constructing overlay networks that satisfy the strong relay-free property using Skip Graph [12]. Subsequently, we describe the mechanism of suspending publishing on the constructed overlay networks.

5.1 Skip Graph

Skip Graph is an algorithm of structured overlay networks providing the function of range search. Each node has a key and can issue a query by specifying a range or a value in the key space. Issued queries are 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 [18], as shown in **Fig. 4**. Level 0 is a doubly linked list that consists of all nodes sorted in the order of keys. Each node has a random sequence in base b*1 called a membership vector, and composes a doubly linked list with nodes whose membership vectors have the same first i digits in level i.

When a node issues a query, the search process starts from the maximum level of the node. The query is forwarded among nodes in the same manner as the skip list, i.e., skips long distance at the higher level and gradually moves down to level 0.

The size of the routing table that each node must have is $O(\log N)$ of the N participants, while the path length of forwarding queries is also $O(\log N)$.

5.2 Multi-key Skip Graph

Multi-key Skip Graph [19] is an expansion of Skip Graph, which enables participating nodes to possess multiple keys. Each node (hereafter, called physical node) inserts its keys onto Skip Graph as virtual nodes. Virtual nodes created from the same physical node have an equivalent membership vector, namely membership vectors are unique to physical nodes.

If a search query is forwarded among virtual nodes in the same way as normal Skip Graph, there is a possibility that the query passes through one physical node multiple times. To avoid an increase in hops by such possibility, Multi-key Skip Graph includes an efficient routing mechanism called multi-range forwarding.

In multi-range forwarding, a query with its target range R is forwarded as follows:

> When a virtual node whose key is outside R receives the query, the virtual node selects one from the virtual nodes of its physical node on the basis of proximity to R, and hands the query over to it. If the nearest is itself, it processes forwarding in the same way as Skip Graph.
>
> When a virtual node whose key is within R receives the query, the virtual node divides R into subranges by the keys of its physical node. The query is duplicated and forwarded to other physical nodes with each subrange attached instead of R. **Figure 5** shows an example. There are three physical nodes whose membership vectors are 00, 01, and 10. When the virtual node, whose key is 0, receives a query of target range $0 \leq key \leq 6$, the range is divided into three subranges: A, B, and C. Subrange B and C are forwarded to physical node 01 from 00, then are divided into subranges: B into B1 and B2, C into C1 and C2. Finally, subrange C2 is forwarded to physical node 10 from 01 and devided into C2α and C2β, but they expire because there are no more physical nodes to receive the query.

With these rules, each physical node receives the same query only once, and the path length of forwarding queries is $O(\log N)$ where N is the number of physical nodes but not virtual nodes.

5.3 Proposed Method

5.3.1 Construction Satisfying Strong Relay-free Property

We first assume that each node possesses the names of topics of interest as a subscriber or a publisher. By using the names as keys and constructing Multi-key Skip Graph, topic-based pub/sub is possible. Publishers can deliver messages to subscribers by range queries of Multi-key Skip Graph. At this point, the overlay network is relay-free because subscribers and publishers joining the same topic are contiguous at level 0.

With our method, we give totally ordered relation between subscribers and publishers with lower priority than topic names. For instance, such a totally ordered relation is possible by adding suffixes that are different among subscribers and publishers to topic names, e.g., $T1\_pub$ is the key of publishers of topic $T1$ while $T1\_sub$ is the key of subscribers. As shown in **Fig. 6**, in addition to the fact that the units of every topic are sorted, units of every node type (subscriber or publisher) are also sorted inside the topic units at level 0.

In this overlay network, subscribers and publishers of each

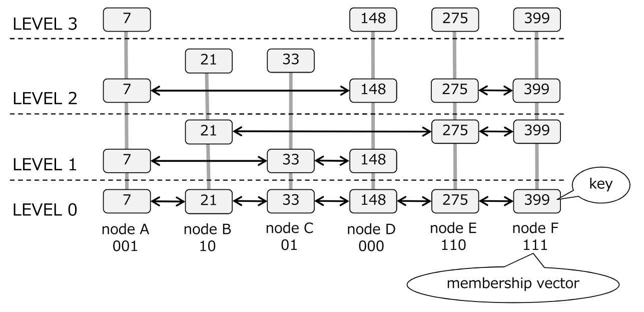


Fig. 4 Example of Skip Graph.

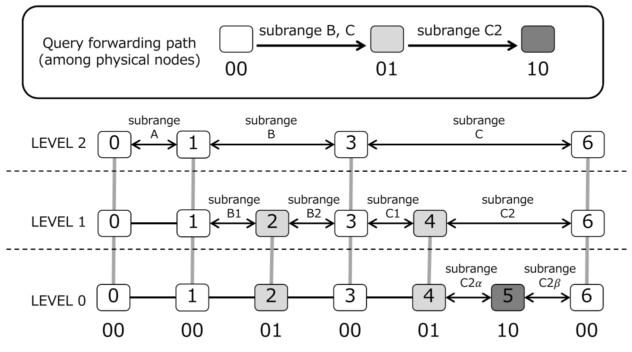


Fig. 5 Query forwarding by multi-range forwarding.

*1 In this paper, we consider the case of binary digits.

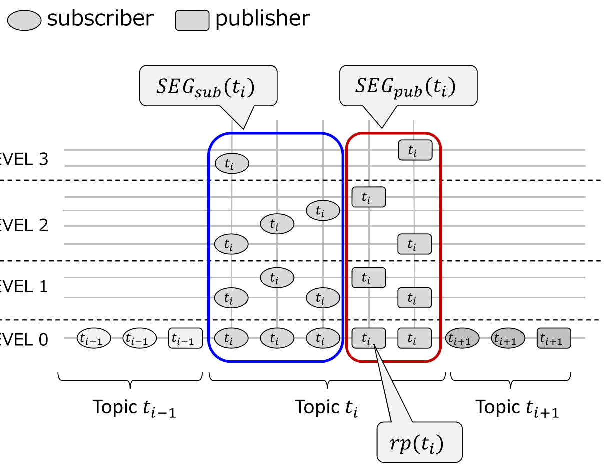


Fig. 6 Ordered relation of nodes in proposed method.

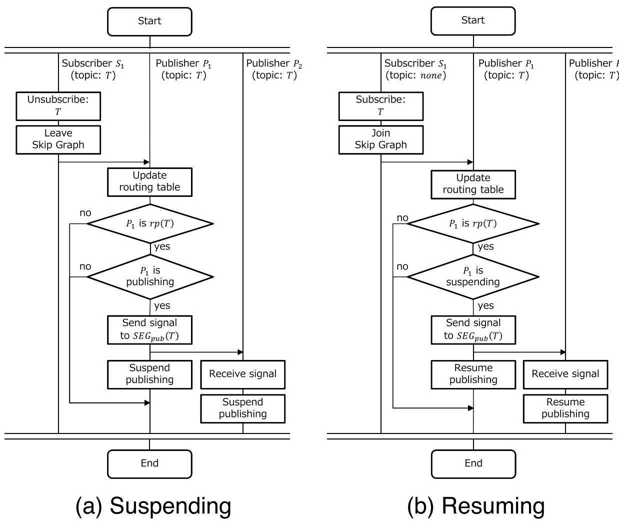


Fig. 7 Flow chart with respect to switching publishers' behavior.

topic are contiguous respectively at level 0, and the subgraph of subscribers and that of publishers are also contiguous. Thus, the overlay network satisfies the strong relay-free property.

Hereafter, we express the subgraph of publishers of topic $t \in T$ as $SEG_{pub}(t)$, and the subgraph of subscribers of topic $t \in T$ as $SEG_{sub}(t)$.

5.3.2 Publish, Subscribe, Unsubscribe

When a publisher sends a message to a topic t, the range search mechanism of Multi-key Skip Graph is used with the target range of $SEG_{sub}(t)$. The process of subscribing/unsubscribing is possible by the insertion/deletion mechanisms of virtual nodes in Multi-key Skip Graph (essentially similar to those of Skip Graph).

5.3.3 Definition of Rendezvous Point

For $\forall t \in T$, it is ensured that there exists a unique publisher contiguous to $SEG_{sub}(t)$ at level 0 as long as one or more publishers exist. We call this publisher a "rendezvous point" with an expression of $rp(t)$. For example, the position of the rendezvous point of topic t_i is illustrated in Fig. 6. $rp(t)$ can conclude whether subscribers are absent without any meta information about topic t. Specifically, if the only neighbor v of $rp(t)$ at level 0 in the direction of $SEG_{sub}(t)$ leads to $Sub(v, t) = true$, there are one or more subscribers. Otherwise, there are no subscribers.

When a new publisher is inserted between $rp(t)$ and $SEG_{sub}(t)$, the new publisher takes the place of the rendezvous point thereafter. On the other hand, when an existing $rp(t)$ leaves topic t and there are other publishers, the neighbor of $rp(t)$ at level 0 in the direction of $SEG_{pub}(t)$ takes over the position of the rendezvous point.

5.3.4 Suspending and Resuming

Suspending and resuming according to the switching between subscribers' absence and presence is possible by the rendezvous point, which is responsible for detecting the switching and notifying other publishers. **Figure 7** shows flow charts regarding the behavior of $rp(t)$.

When all subscribers of topic t leave, $rp(t)$ can detect it passively by using a handler which catches the update of routing tables of Multi-key Skip Graph. When $rp(t)$ detects the absence of subscribers, it sends a signal dictating suspension to publishers, by using the range search mechanism with the target range of $SEG_{pub}(t)$, as described in Fig. 7 (a).

Conversely, when new subscribers appear in topic t, which has had no subscribers, $rp(t)$ can detect it passively in the same way and is responsible for sending a signal dictating resuming to publishers (see Fig. 7 (b)).

5.3.5 Inserting Publishers

Newly joining publishers need to conclude whether they should start publishing immediately after finishing participation[*2].

In case that there has been any publisher of the corresponding topic, the joining publisher is certain to exchange messages with at least one existing publisher in the process of inserting in Multi-key Skip Graph. In the proposed method, information about the suspending status is piggy-backed on the messages, and the joining publisher determines the correct status by checking it.

If there were no publishers, the joining publisher will be the rendezvous point. Hence, it can determine the correct status by itself after finishing insertion.

Figure 8 is a flow chart regarding node insertions including the above mechanisms.

5.3.6 Eliminating Inconsistencies

In the proposed method, two types of inconsistencies described below can occur by the undelivered signals from rendezvous points caused by the sudden disappearance of nodes on the notifying path.

(i) There exists a publisher continuing to publish even if there are no subscribers.

(ii) There exists a publisher suspending even if there are subscribers.

These inconsistencies can occur on every publisher, except for rendezvous points. We describe the solutions for the inconsistencies.

In i, a published message from a publisher of topic t is certain to pass through $rp(t)$ as long as there are no subscribers. When

[*2] Concerning a node which is both a subscriber and publisher of a topic, such node just needs to insert a virtual node only into $SEG_{sub}(t)$ and continue publishing towards $SEG_{sub}(t)$. The reason is that there exist both subscribers and publishers as long as the node itself is alive. Therefore, such node can be irrelevant to suspending/resuming mechanisms.

Table 1 Qualitative comparison of methods.

	Relay-free	Strong relay-free	Path length	Storage cost
Proposed	✓	✓	$O(\log(pub_t + sub_t))$	$O(\frac{pub_t+sub_t}{N} M \log N)$
Scribe/Bayeux	✗	✗	$O(\log N)$	$O(\frac{pub_t}{N} M + \frac{sub_t}{N} M \log N)$
CAN-MC	✓	✗	$O(d(pub_t + sub_t)^{1/d})$	$O(\frac{pub_t+sub_t}{N} Md + d)$

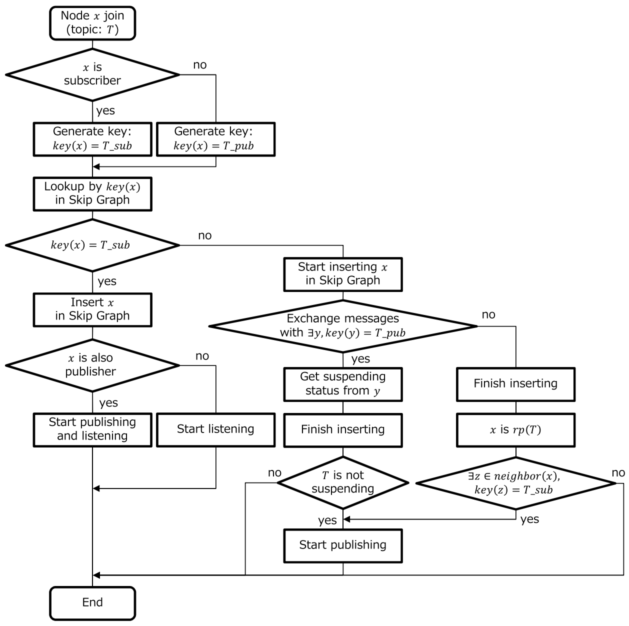

Node x join (topic: T)
x is subscriber
no
yes
Generate key: $key(x) = T\_sub$
Generate key: $key(x) = T\_pub$
Lookup by $key(x)$ in Skip Graph
$key(x) = T\_sub$
no
yes
Start inserting x in Skip Graph
Insert x in Skip Graph
Exchange messages with $\exists y, key(y) = T\_pub$
no
yes
x is also publisher
no
yes
Get suspending status from y
Finish inserting
Start publishing and listening
Start listening
Finish inserting
x is $rp(T)$
T is not suspending
no
yes
$\exists z \in neighbor(x), key(z) = T\_sub$
no
yes
Start publishing
End

Fig. 8 Flow chart of node insertion.

$rp(t)$ receives a message from other publishers during suspend, $rp(t)$ checks the absence of subscribers and sends a signal dictating suspension to the source publisher.

In ii, on the other hand, each publisher that is suspending actively confirms the status concerning the suspension of the neighbor at level 0 by periodically sending a dedicated message. As a result of the confirmation, if there is a conflict between the status of the neighbor and itself, the publisher sends a reporting message towards $rp(t)$. When $rp(t)$ receives the report, it checks the presence of subscribers and sends a signal dictating resuming to $SEG_{pub}(t)$.

5.4 Qualitative Assessment

We give a qualitative assessment of the proposed method in comparison with the methods described in Section 3.1[*3]. We assume the following notations: M denotes the number of topics, N denotes the number of nodes, pub_t denotes the number of publishers per topic, sub_t denotes the number of subscribers per topic, and d denotes the number of dimensions in CAN. To simplify, we also assume that each node is not both a subscriber and publisher of the same topic.

Table 1 shows the comparison of the methods. The proposed method and CAN-MC both satisfy the relay-free property. This characteristic brings about good features of path length described later.

Strong relay-free property is satisfied only by the proposed method. It means that the proposed method can suspend publishing as described above, while the other methods cannot.

"Path length" denotes the maximum length of paths from publishers to subscribers. The proposed method needs $O(\log(pub_t + sub_t))$, because a published message of topic t is forwarded over the subgraph which consists of $SEG_{sub}(t)$ and $SEG_{pub}(t)$. Scribe/Bayeux uses lookup paths of DHTs, so the path length is $O(\log N)$. CAN-MC requires $O(d(pub_t+sub_t)^{1/d})$, due to flooding over the corresponding mini CAN. Because the path length of the proposed method does not depend on N unlike Scribe/Bayeux, it can reduce the consumption of network resources and the delay time of delivery, especially regarding topics having a small number of participants. CAN-MC also excludes N, and its path length depends on d which can adjust the tradeoff between the path length and storage cost.

"Storage cost" denotes the average size of routing tables of all nodes. This cost affects the consumption of memory and the maintenance overhead on each node. With the proposed method, each publisher or subscriber is inserted onto Multi-key Skip Graph as a virtual node which must have $O(\log N)$ neighbors in the routing table. Thus, the average size of routing tables is $O(\frac{pub_t+sub_t}{N} M \log N)$. Regarding Scribe/Bayeux, each subscriber forces intermediate nodes on the forwarding path to possess children tables. This storage cost is $O(\frac{sub_t}{N} M \log N)$ on average. Besides this, each publisher caches the root node of the corresponding topic, then the average cost is $O(\frac{pub_t}{N} M)$. Accordingly, the total average cost is $O(\frac{pub_t}{N} M + \frac{sub_t}{N} M \log N)$. With CAN-MC, each publisher or subscriber is inserted onto mini CAN with the cost of $O(d)$, so the average cost is $O(\frac{pub_t+sub_t}{N} Md)$. Each node also composes the entire CAN, thus $O(\frac{pub_t+sub_t}{N} Md + d)$ is required as a whole. The proposed method requires slightly large cost compared to Scribe/Bayeux, but is not extremely inferior. Concerning CAN-MC, the cost depends on d.

There is another viewpoint that should be discussed. Node(s) that are responsible for the storage cost are different between the methods. Regarding the proposed method and CAN-MC, when a subscriber or publisher is added, the joining subscriber or publisher itself is responsible for the storage cost. Some other nodes are forced to update routing tables, but no one is basically forced to increase the size of its routing table except for the joining node. On the other hand, with Scribe/Bayeux, the joining of a subscriber or publisher forces intermediate nodes on the forwarding path to take responsibility of the storage cost. This seems to be preferable from the viewpoint of load distribution, but it also means that each node cannot predict its forwarding load. In other words, the fact that multiple nodes are responsible for storage cost may lead to inconvenience in terms of the load predictability. The details will be discussed later with experimental results, in Section 6.3.

[*3] In this section, we refer to Scribe, Bayeux, and CAN-MC. As for DYNATOPS, the fundamental property is considered to be same as Scribe and Bayeux.

6. Evaluation

We evaluated our method through experiments with a simulation program implemented in Java. This section gives the details of each experiment and its results. We chose Scribe as the comparison target in these experiments, mainly because it can be built on top of any DHTs. Skip Graph can be used to construct a DHT by using a kind of routing which is referred to as "Routing by Numeric ID" in SkipNet [20]. Indeed, this type of DHT is implemented by PIAX [21]. Using a DHT on top of Skip Graph is convenient for harmonizing the experimental conditions such as the size of routing tables with the proposed method[*4]. Therefore, we implemented Scribe on top of Skip Graph in the simulation program.

Note that our experiments were aimed to show essential tendencies because the actual performance is affected by parameters, e.g., the radix of membership vectors can adjust the tradeoff between the path length and size of routing tables.

6.1 Number of Messages Associated with Publishing

To confirm the ability to suspend publishing, we measured the number of forwarded messages on the overlay network with several numbers of subscribers including zero. The simulator generated 100,000 nodes and constructed an overlay network regarding the proposed method and Scribe respectively. We set a topic with the following two patterns:

- A topic has 100 publishers and 1,000 subscribers.
- A topic has 10 publishers and 1,000 subscribers.

The simulator made subscribers unsubscribe in turns. At the timing of that the number of subscribers matches 1,000, 100, 10 and 0, the simulator forced publishers of the topic to publish a message and counted the number of messages forwarded on the overlay network.

Figure 9 shows the results of the averages of five repeated measurements. The term "p/t" in the figure denotes the number of publishers per topic. The graph indicates that the number of messages drops to 0 when the number of subscribers is 0 regarding both patterns of the proposed method. Regarding Scribe, messages are forwarded even if the number of subscribers is 0. With both methods, the number of messages becomes large according to the number of publishers or subscribers. When the number of subscribers is 1,000, 100 or 10, the number of messages of the proposed method is smaller by approximately one digit than Scribe. This result is due to the difference of the length of forwarding path. In the proposed method, the length depends on the number of participants of the corresponding topic. On the other hand, the length in Scribe depends on the number of whole nodes, thus longer paths which cause a lot of messages are constructed. Experiments for confirming the difference of the path length will be described in Section 6.2.

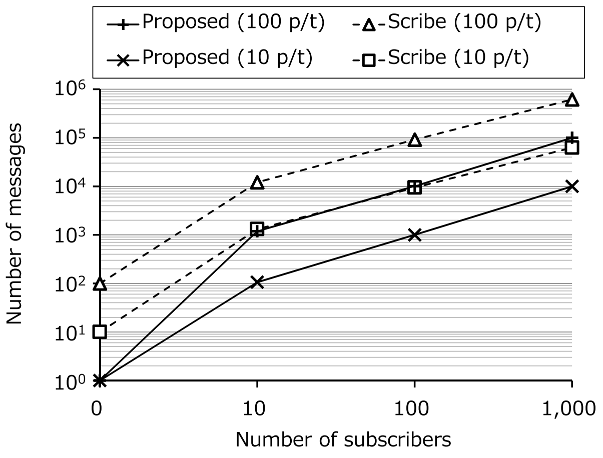


Fig. 9 Number of messages associated with publishing.

Table 2 Patterns of the experiment.

	pub_t	sub_t	Number of total nodes
α	10	990	1,000 or 10,000 or 100,000
β	500	500	1,000 or 10,000 or 100,000
γ	990	10	1,000 or 10,000 or 100,000
δ	1	9	1,000 or 10,000 or 100,000
ϵ	5	5	1,000 or 10,000 or 100,000
ζ	9	1	1,000 or 10,000 or 100,000

6.2 Length of Forwarding Path of Publishing

We also evaluated the effectiveness against gratuitous forwarding mentioned in Section 3.2. In this experiment, we calculated the average length of paths from each publisher to each subscriber. Here pub_t and sub_t denote the same as described in Section 5.4.

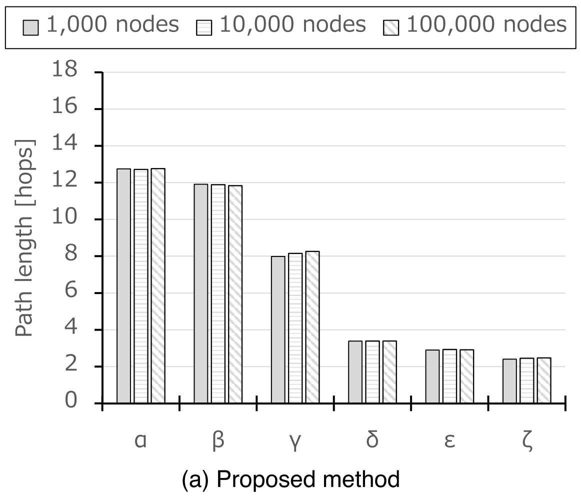
In this experiment, every topic was the same size, i.e., the sum of the number of publishers and subscribers was equivalent. We assumed two topic-sizes: small and large. We also assumed three combinations of pub_t and sub_t: $pub_t < sub_t$, $pub_t = sub_t$ and $pub_t > sub_t$. Thus, we set six patterns in total, as listed in **Table 2**. Each pattern had three different node amounts, 1,000, 10,000 and 100,000. The number of topics in each pattern was keyed to the number of nodes, i.e., it can be obtained by dividing "Number of total nodes" by the sum of pub_t and sub_t. For example, the number of topics in pattern α was 10 when the number of nodes was 10,000.

The simulator constructed overlay networks for every pattern and calculated the average length of the forwarding path from a publisher to every corresponding subscriber for all publishers.

Figure 10 shows the results. Regarding the proposed method, Fig. 10 (a) illustrates that the path length was not affected by the total number of nodes and was decreased in response to the reduction of the size of topics. This means the proposed method has high scalability for the increase in the total number of nodes and can prevent gratuitous forwarding. On the other hand, the results for Scribe shown in Fig. 10 (b) indicate that the path length is not affected by the size of topics, which causes gratuitous forwarding.

For example, when focusing on patterns of δ, ϵ and ζ with 100,000 nodes, the path length is less than 4 hops in the proposed method while Scribe requires more than four times the hops (16 hops). The path length affects the latency between publishers and

[*4] Bayeux can also be built on top of any DHTs, unlike that CAN-MC is specialized for using CAN. But Bayeux has almost the same characteristics as Scribe from the viewpoint of the experiments described in this section, i.e., the number of messages, the length of the forwarding path and the correlation between the number of sending/receiving and forwarding messages. Therefore, we have chosen Scribe which was proposed later than Bayeux.



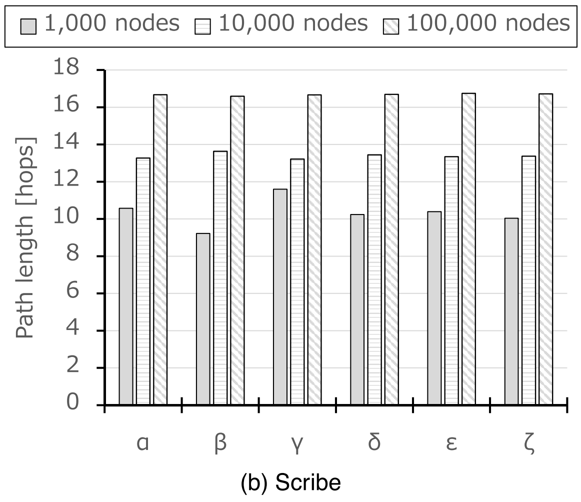


Fig. 10 Length of forwarding path of publishing.

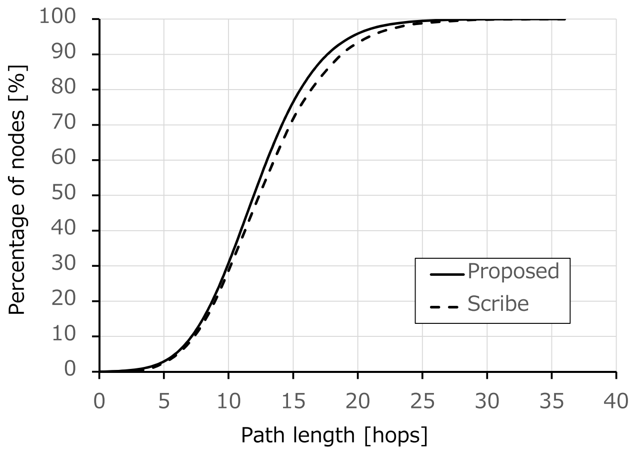


Fig. 11 CDF of path length of pattern α.

subscribers, and also the number of messages as shown in Section 6.1.

Focusing on the pattern α of which the path lengths are most similar between the proposed method and Scribe, we also confirmed the cumulative distribution function (CDF) of path length as shown in **Fig. 11**. It can be seen that there was no significant difference between the two curves, i.e., the above convenient features of the proposed method do not cause a serious undesirable effect on the distribution of path length.

6.3 Correlation between Number of Sending/receiving and Forwarding Messages

We focused on the correlation between the number of sending/receiving and forwarding messages. This viewpoint is important for predicting the load of each node, namely edge broker.

In distributed pub/sub using structured overlay networks, each node is responsible for forwarding messages to relevant succeeding destinations. The forwarding load is determined according to properties of topics which the node is on the paths of. For example, the load must be heavy regarding a node responsible for forwarding messages of a topic whose publishers frequently send large amounts of data. The forwarding load is closely related to routing tables, which store the forwarding path information on each node. The information is registered on nodes in a different way for every method, as described in Section 5.4.

If the forwarding load correlates with the transmission load as publishers or receiving load as subscribers, each edge broker can easily predict the necessary specifications of hardware resources. For instance, if there is a device attempting to subscribe to a topic of video streaming, an edge broker which the device joins will be under a heavy load and should be strengthened.

Conversely, if the forwarding load does not correlate, it is difficult to predict from local information. Such a case is unsuitable when it is assumed that edge brokers compose autonomous distributed networks such as the Internet, namely edge brokers are not managed by a single enterprise intensively but are arbitrarily added/removed by various enterprises or individuals.

In this regard, we conducted an experiment in which the simulator counts the number of forwarding and sending/receiving messages for every node. Precisely, "number of forwarding" is the number of times that a physical node forwards a message to others, including the initial hops from publishers. "number of sending" is the number of times that a publisher sends a message created on itself. "number of receiving" is the number of times that a subscriber receives a message associated with the topic the subscriber is subscribing to.

The conditions of this experiment are as follows: pub_t was 1 and sub_t was 1,000. The number of topics was 100, thus the total number of nodes was 100,100. The 100 publishers joining different topics were assigned different time intervals of sending. The intervals were calculated so as to make the number of transmissions during the simulation period become $1, 2, \ldots, 100$.

The simulator constructed overlay networks with the above conditions, and forced publishers to publish at respective intervals. After completion of counting the number of forwarding and sending/receiving, we normalized the data. The normalizing function for a data x in a data set X is as follows:

$$Normalize(x) = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Figure 12 shows the results obtained by plotting all publishers, and **Fig. 13** shows those by plotting 1,000 subscribers which are randomly selected from all of subscribers. Regarding the proposed method, both the number of sending and receiving messages are clearly correlated with the number of forwarding messages. In Fig. 13, nodes of the proposed method are plotted linearly on three different angled lines. This is because of the char-

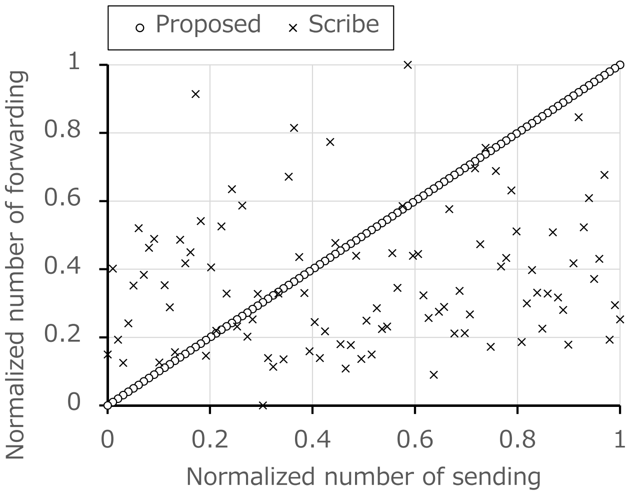


Fig. 12 Correlation between number of sending and forwarding.

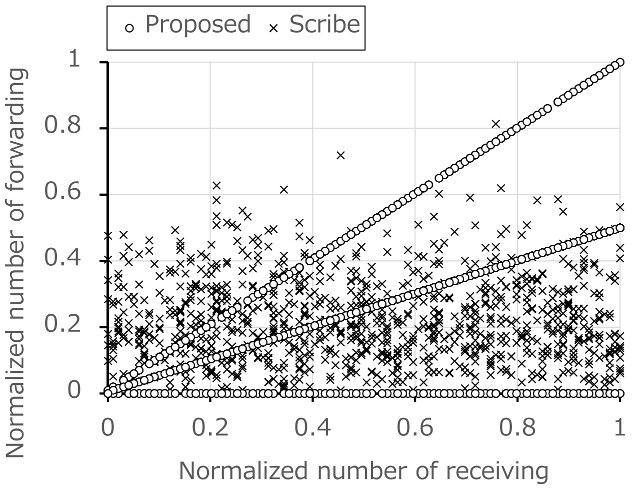


Fig. 13 Correlation between number of receiving and forwarding.

Table 3 Correlation coefficients and confidence intervals.

	Correlation coefficients	99% Confidence intervals
Figure 12 - proposed	1.0	N/A
Figure 12 - Scribe	0.1310	$-0.1290 \le \rho \le 0.3742$
Figure 13 - proposed	0.5483	$0.5426 \le \rho \le 0.5540$
Figure 13 - Scribe	-0.0045	$-0.0126 \le \rho \le 0.0037$

acteristic of multi-range forwarding in Multi-key Skip Graph, which forces each node to forward at most twice for each dissemination of a published message[*5]. In contrast, the results of Scribe indicate that there are no correlations.

The correlation coefficients and their confidence intervals at the 99% level were calculated[*6], as shown in **Table 3**. Note that the confidence interval of proposed method in Fig. 12 is written as N/A because the Fisher transformation cannot be applied on the correlation coefficient value of 1.0.

Considering practical applications, it is natural that the frequency of publishing is unbalanced. For example, Twitter is a famous and large service based on pub/sub messaging. It has been reported that the number of tweets for every user follows the power law distribution, and 20% of users account for 84% of tweets [22]. Scribe or similar methods receive a negative effect from such an imbalance in terms of load predictability. In fact, there was a node that was forced to forward more than one thousand times the number of receiving count regarding Scribe in the experiment. In contrast, nodes in the proposed method forwarded at most twice the number of receiving count.

*5 Specifically, the forwarding paths in Multi-key Skip Graph compose incomplete binary trees. Each root node or intermediate node has one or two children, and each leaf node has no child. This is why each node forwards a message at most twice.

*6 We used original data before normalization. Regarding Fig. 13, we used all of the data, not sampled data.

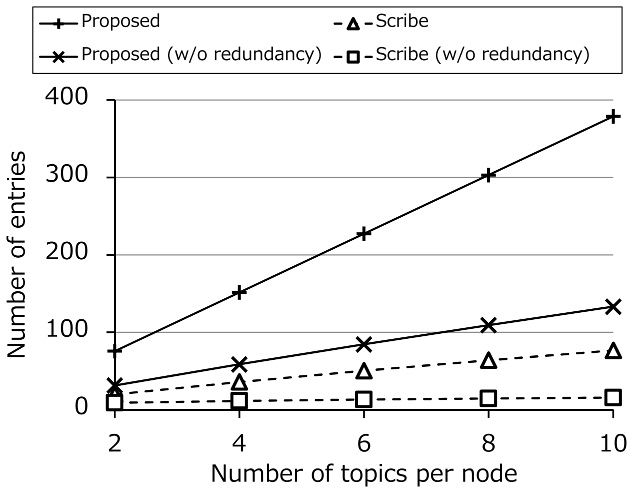


Fig. 14 Size of routing tables.

6.4 Size of Routing Tables

Furthermore, we conducted experiments for observing the size of routing tables on each node. Routing table size affects maintenance cost including the consumption of memory space.

At first, we focused on the transition of the average size of routing tables in response to the number of topics which each node publishes or subscribes to. The simulator constructed overlay networks with 100,000 nodes. Half of them joined as publishers, and the remaining half joined as subscribers. Each node published/subscribed to specific number of topics uniformly: 2, 4, 6, 8 and 10. The topics are randomly selected from predefined 1,000 topics. For each number of topics, the simulator calculated the average number of entries in routing tables of each node.

Figure 14 shows the result. For each method ("Proposed" or "Scribe"), the simulator counted two kinds of numbers: allow or disallow redundancy. In the legend of the graph, "w/o redundancy" denotes that the simulator counted the number of unique physical nodes in routing tables. The other data series which "w/o redundancy" is not attached to show the results of counting the number of entries in routing tables by allowing redundancy. The number of unique physical nodes in routing tables affects a kind of maintenance cost, when each node actively checks the existence of neighbors, i.e., periodically sends messages to them for confirming their activity.

In the Figure, the absolute values of the vertical axis of the proposed method are larger than Scribe, but the values do not increase intensively because they change linearly with the number of topics just like Scribe. Note that the proposed method based on Skip Graph can reduce the size of routing tables with a trade-

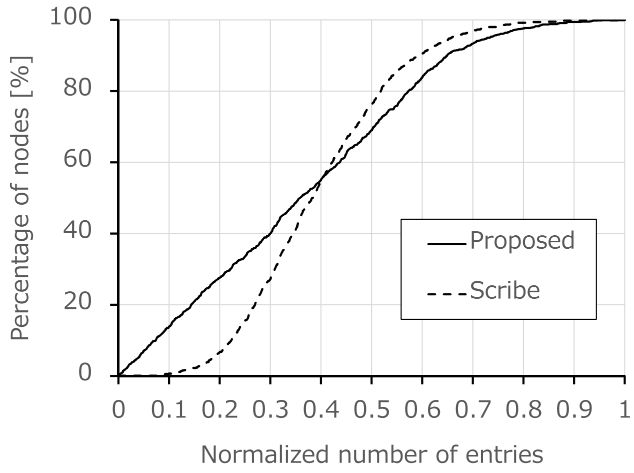


Fig. 15 CDF of size of routing tables.

off of the increase of the path length, by configuration of the base number of a logarithm.

Subsequently, we focused on the CDF of the size of routing tables, to confirm the difference of the responsibility for the storage cost which is described in Sections 5.4 and 6.3. The simulator constructed overlay networks with 1,000 nodes, which all joined as subscribers. The number of topics to which each node subscribed was decided by selecting one from 1 to 1,000 without duplication. The topics on each node were different one by one. The simulator counted the number of entries in routing tables of each node by allowing redundancy.

The obtained data were normalized by the function described in Section 6.3, and plotted as shown in **Fig. 15**. It can be seen that the percentage of nodes in the proposed method is linearly increased compared to Scribe. This is caused by a characteristic by which the storage cost of a node is sensitive to the number of topics to which the node subscribes. On the other hand, in Scribe, the storage cost is shared among nodes. It leads to difficulty regarding load predictability as described in Section 5.4.

7. Conclusion

For handling exhaust data in topic-based pub/sub messaging, we presented a novel design of overlay networks by defining "strong relay-free" property. It enables to convert the problem of detecting the presence/absence of subscribers into the problem of detecting those of one subgraph.

Subsequently, we proposed a method using Skip Graph which can construct overlay networks that satisfy the above property. The proposed method is highly scalable and can suspend publishing by detecting the absence of subscribers and prevent the gratuitous forwarding of published messages.

From the simulation experimental results, we confirmed that the above characteristics work effectively with the proposed method in comparison with Scribe. Regarding the problem of gratuitous forwarding, the path length of the proposed method was less than one fourth that of Scribe when the number of nodes was 100,000. It was also shown that the proposed method could predict the forwarding load, which Scribe could not.

These results indicate that our method is suitable for the edge broker model described in Section 2. The growth in IoT accelerates the creation of exhaust data. Therefore, the proposed method can reduce the wasting of network resources and encourage the locally produced data to be consumed locally. The proposed method can be useful for not only the edge broker model but also various situations with a large amount of nodes, e.g., pub/sub messaging in a single data center.

One topic for future work is to reduce the size of routing tables, which tend to be larger than existing methods as described in Section 6.4. Furthermore, we intend to do the more practical evaluation. This time we focused on confirming the characteristics of the proposed method. We believe it is also important to evaluate the actual effectiveness of it. For example, the effect of suspending is considered to depend on the distribution of the number of subscribers along the time axis direction. Thus, we plan to use actual data that reflect various biases of distributions in the real world, e.g., the relation data of SNSs. We also plan to evaluate the proposed method on actual networks.

References

[1] Hodges, S., Taylor, S., Villar, N. and Scott, J.: Prototyping Connected Devices for the Internet of Things, *IEEE Computer*, pp.26–34 (2013).

[2] Eugster, P.T., Felber, P.A., Guerraoui, R. and Kermarrec, A.-M.: The Many Faces of Publish/Subscribe, *ACM Computing Surveys*, Vol.35, No.2, pp.114–131 (2003).

[3] Rowstron, A. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp.329–350 (2001).

[4] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: A scalable Content-Addressable Network, *ACM SIGCOMM*, Vol.31, No.4, pp.161–172 (2001).

[5] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D. and Kubiatowicz, J.D.: Tapestry: A resilient global-scale overlay for service deployment, *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, pp.41–53 (2004).

[6] Castro, M., Druschel, P., Kermarrec, A.-M. and Rowstron, A.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure, *IEEE Journal on Selected Areas in Communications*, Vol.20, No.8, pp.1489–1499 (2002).

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

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

[9] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. and Byers, A.H.: *Big data: The next frontier for innovation, competition, and productivity*, McKinsey Global Institute (2011).

[10] Chockler, G., Melamed, R., Tock, Y. and Vitenberg, R.: Constructing Scalable Overlays for Pub-Sub with Many Topics, *ACM Symposium on Principles of Distributed Computing*, pp.109–118 (2007).

[11] Setty, V., Steen, M.V., Vitenberg, R. and Voulgaris, S.: PolderCast: Fast, Robust, and Scalable Architecture for P2P Topic-based Pub/Sub, *International Middleware Conference*, pp.271–291 (2012).

[12] Aspnes, J. and Shah, G.: Skip Graphs, *ACM Trans. Algorithms* (*TALG*), Vol.3, No.4, pp.37:1–37:25 (2007).

[13] Oracle: Java Message Service (JMS), available from ⟨www.oracle.com/technetwork/java/jms⟩ (accessed 2014-01-31).

[14] OASIS: AMQP, available from ⟨www.amqp.org⟩ (accessed 2014-01-31).

[15] NTT: Edge computing, available from ⟨www.ntt.co.jp/news2014/1401e/140123a.html⟩ (accessed 2014-01-31).

[16] Zhang, R. and Hu, Y.C.: Borg: A Hybrid Protocol for Scalable Application-level Multicast in Peer-to-Peer Networks, *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp.172–179 (2003).

[17] Zhao, Y., Kim, K. and Venkatasubramanian, N.: DYNATOPS: A Dynamic Topic-based Publish/Subscribe Architecture, *International Conference on Distributed Event Based Systems*, pp.75–86 (2013).

[18] Pugh, W.: Skip Lists: A Probabilistic Alternative to Balanced Trees, *Comm. ACM*, Vol.33, No.6, pp.668–676 (1990).
[19] Konishi, Y., Yoshida, M., Takeuchi, S., Teranishi, Y., Harumoto, K. and Shimojo, S.: An Extension of Skip Graph to Store Multiple Keys on Single Node, *Journal of Information Processing Society of Japan*, Vol.49, No.9, pp.3223–3233 (2008) (in Japanese).
[20] Harvey, N.J.A., Dunagan, J., Jones, M.B., Saroiu, S., Theimer, M. and Wolman, A.: SkipNet: A Scalable Overlay Network with Practical Locality Properties, *USENIX Symposium on Internet Technologies and Systems*, pp.9–23 (2003).
[21] PIAX: PIAX: P2P Interactive Agent eXtensions, available from ⟨www.piax.org/en⟩ (accessed 2014-01-31).
[22] Welhuis, A.: Twitter and the pareto principle, available from ⟨www.annouckwelhuis.nl/twitter-and-the-pareto-principle-2⟩ (accessed 2014-01-31).

Ryohei Banno received his B.E. in Information Engineering in 2010, and his Master of Information Science and Technology in 2012, all from Hokkaido University, Japan. Since 2012, he has been a researcher in NTT Network Innovation Laboratories. His research interests include distributed systems, especially structured overlay networks. He is a member of IPSJ, IEICE, and JSSST.

Susumu Takeuchi received his M.E. and Ph.D. degrees from Osaka University, Japan, in 2003 and 2006, respectively. From 2006 to 2009, he was an assistant professor of Graduate School of Information Science and Technology, Osaka University. In 2009, he joined National Institute of Information and Communications Technology (NICT) as an expert researcher. Since 2011, he has been a senior research engineer in NTT Network Innovation Laboratories. He received the IPSJ Best Paper Award in 2011. His research interests include technologies for information systems, especially utilizing social networks and overlay networks. He is a member of IPSJ and IEEJ.

Michiharu Takemoto received his B.S. and M.S. degrees from the University of Tokyo, Tokyo, Japan, in 1992 and 1994, respectively. He received his Ph.D. in Engineering from Waseda University, Tokyo, Japan in 2009. Since he joined NTT in 1994, his research interests include distributed computing environment and its applications. He received a Young Investigators Award from IEICE in 1998. He was a visiting scientist at Laboratory for Computer Science, Massachusetts Institute of Technology from 1999 to 2000. He was a guest researcher at National Institute of Information and Communications Technology (NICT) of Japan from 2008 to 2011. He is currently a senior research engineer and supervisor, in NTT Network Innovation Laboratories. He is a member of IEEE, IPSJ and IEICE.

Tetsuo Kawano received his B.E. degree from Kumamoto University, Kumamoto, Japan in 1991. He received his M.E. degree and Ph.D. in Engineering from Kyushu University, Fukuoka, Japan in 1993 and 1996, respectively. He joined NTT Software Laboratories, Tokyo, Japan in 1996. Currently, he works as senior research engineer in NTT Network Innovation Laboratories, where he is studying distributed network service platform architectures. He is a member of IPSJ and IEICE.

Takashi Kambayashi received his B.S. and M.S degrees from Keio University in 1987 and 1989, respectively. Since he joined NTT in 1989, his interests include ubiquitous networking systems. He is a Manager in NTT Science and Core Technology Laboratory Group. He is a member of IPSJ.

Masato Matsuo received B.E. and M.E. degrees in Mechanical Engineering from Kyoto University in 1986 and 1988, respectively. After joining NTT in 1988, he researched adaptive network service systems and ubiquitous networking systems. He is a senior research engineer and supervisor in NTT Network Innovation Laboratories. He is a member of IPSJ and IEICE.