A Distributed Topic-based Pub/Sub Method for Exhaust Data Streams Towards Scalable Event-driven Systems

Ryohei Banno, Susumu Takeuchi, Michiharu Takemoto, Tetsuo Kawano, Takashi Kambayashi and Masato Matsuo
NTT Network Innovation Laboratories, NTT Corporation
Tokyo, Japan
Email: banno.ryohei@lab.ntt.co.jp

***Abstract*—Distributed pub/sub messaging has become indispensable for event-driven systems. There are methods for achieving high scalability regarding topic-based pub/sub by using structured overlay networks. However, these methods waste network resources concerning "exhaust data," which have low or no value most of the time. There are at least two problems: each publisher node continues to forward data to a relay node even if there are no subscribers, and multicast trees are constructed which are excessively large for low value data, namely having a small number of subscribers. In this paper, we formulate the requirements of overlay networks by defining a property called "strong relay-free" as an expansion of relay-free property, and propose a practical method satisfying the property by using Skip Graph. The proposed method involves publishers and subscribers composing connected subgraphs to enable detecting the absence of subscribers and autonomously adjusting the tree size. 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. The proposed method is competent for decentralized event-driven systems with encouraging the locally produced data to be consumed locally.**

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

I. Introduction

The spread of Internet-connected devices has lead to a variety of discussions about the coming of the Internet of Things (IoT) [1]. The IoT is expected to encompass 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 provide such services, pub/sub messaging [2] is indispensable due to the efficiency for real-time delivery of sensor data compared to traditional request-reply messaging. In topic-based pub/sub, messages are published to logical channels called "topics". Users subscribe to topics of interest and receive messages published on those topics. This paradigm in which messages are exchanged through topics provides decoupling between publishers and subscribers, e.g., each publisher has no concern with the location of subscribers that will receive a published message.

Conventional topic-based pub/sub architectures have a broker server for managing topics [3], [4]. The broker gathers all published messages and processes filtering and forwarding 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 a certain kind of data called "exhaust data", which is predicted to occupy most of the IoT data [5]. The characteristics of exhaust data can be described as follows:

- Wide area

  Data are generated over a wide area in the physical space.
- High frequency

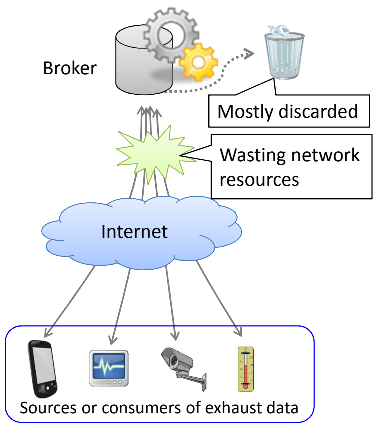
  Data are automatically and continuously generated by devices, unlike today's Internet in which humans generate most of the content.
- Low value density

  Data are generated as byproducts and without specific uses. These data have low or no value most of the time, but sometimes highly useful such like drive recorders.

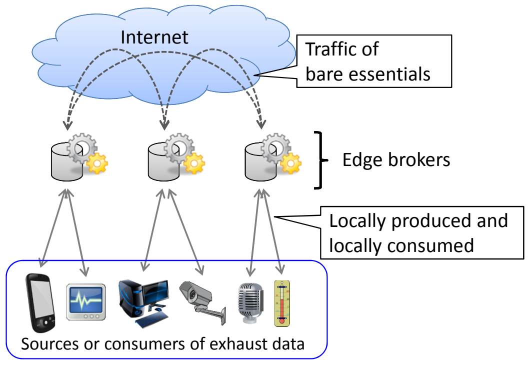
Namely, a tremendous amount of published data is concentrated on the broker with oppressing network resources. This is unprofitable because the data arriving at the broker are mostly discarded due to its low value density (see Figure 1(a)).

Accordingly, we are focusing on a model using edge brokers as shown in Figure 1(b). Edge brokers are placed in front of the Internet described as a cloud in the figure, i.e., they are installed over a wide area. Published data and subscriptions are collected at the closest edge broker, and edge brokers exchange essential data. This model prevents imprudent forwarding of exhaust data to the Internet and encourages the locally produced data to be consumed locally. 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" [6].

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


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 produced and locally consumed
Sources or consumers of exhaust data

(b) Edge broker model

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

video stream to devices joining other edge brokers through the Internet. Possibly these events lead to a flash crowd, which needs scalable multicast mechanisms.

Our aim with this research is to enable pub/sub messaging between edge brokers for exhaust data streams 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.

The number of edge brokers should be large, therefore, a scalable method for managing pub/sub messaging is necessary. We focus on structured overlay networks known primarily for Distributed Hash Tables (DHTs) [7], [8], [9]. 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 [10], [11], [12].

However, these methods are not suitable for the edge broker model because each broker must forward the published data to other broker(s) even if there are no subscribers. Furthermore, some of the methods construct a multicast tree for each topic without taking into account the number of participants (i.e., subscribers or publishers) of the topic. Thus, in the case of topics that have a small number of participants, the published data are forced to be gratuitously forwarded along the excessively large trees. This wastes network resources and increases delay time, though the trees are effective for topics that have a large number of participants. These are serious problems because exhaust data have low value densities, i.e., situations in which the number of participants is zero or small are common.

For overcoming these inefficiencies, we first clarify the requirements of overlay networks and focus on a property called "relay-free", which is mainly known in studies based on unstructured overlay networks [13], [14]. The satisfaction of the property provides an effectiveness for preventing the gratuitous forwarding. However, it is still difficult to suspend publications when there are no subscribers. Therefore, we newly define a desirable property called "strong relay-free" as an expansion of relay-free. In the strong relay-free property, publishers and subscribers compose connected subgraphs respectively, and these subgraphs are also connected to each other. This leads to detectability of the absence of subscribers and autonomous adjustability of the tree size depending on the number of subscribers and publishers.

Subsequently, we propose a method for constructing overlay networks satisfying the property using Skip Graph [15]. We implemented simulation programs of the proposed method and one of the DHT-based methods for some experiments with up to approximately $100,000$ nodes. We compared the ability of suspending publications and autonomous adjustability of tree size, and confirmed the advantage of the proposed method. The experimental results also show that our method can predict the load on each edge broker unlike with the conventional method.

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

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

A. *Topic-based pub/sub method using DHTs*

Scribe [10] is an algorithm for achieving topic-based pub/sub and uses the Pastry network [7]. 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

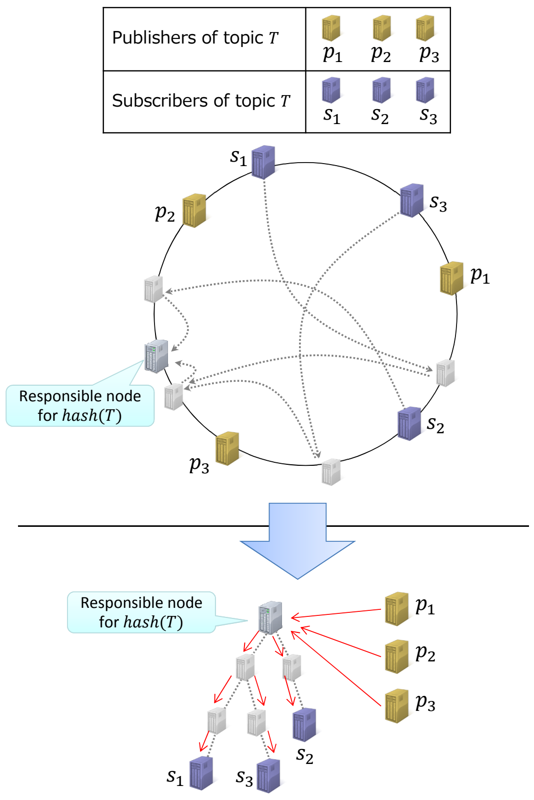


Fig. 2. Topic-based pub/sub by Scribe

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 reverse order of the Pastry's routing path as shown in Figure 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 [11] is built on top of Tapestry [9] 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 [12], built on top of CAN [8], 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.

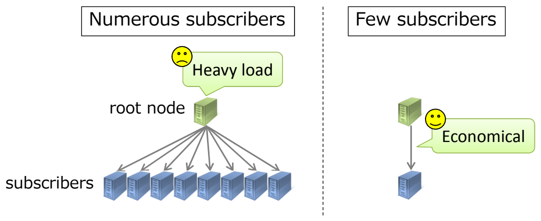
B. Inadequacy for exhaust data streams

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 I. However, conventional methods have the following inefficiencies for handling exhaust data streams on the edge broker model, though they achieve high scalability.

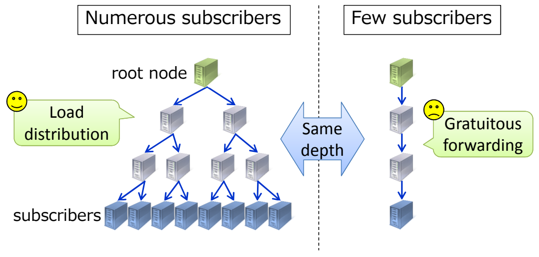
1) Inability to suspend publishing: Conventional methods cannot suspend publishing even if there are no subscribers. In Scribe and Bayeux, publishers have no way to detect the absence of subscribers, and have to continue to constantly send messages towards the root node of the corresponding topic. In 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.

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 $\mathcal{O}(\log N)$ 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. 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 on the left side of Figure 3(a). Scribe and Bayeux construct multicast trees to avoid this heavy load, as shown on the left side of Figure 3(b). 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 right side of Figure



(a) Centralized delivery



(b) Distributed delivery

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

3(b)). Regarding the case of few subscribers, the centralized delivery described in the right side of Figure 3(a) 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.

III. Formulation of Requirements

We first clarify the requirements for overcoming the problems described in Section II-B.

- Against the inability to suspend publishing, it is required that all publishers should detect the switching between subscribers' absence and presence.
- Against 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 primitive ways, such as broadcasting queries to determine the presence of subscribers, switching delivery mechanisms as shown in Figure 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.

A. Relay-free property

The property of relay-free [13], also called Topic-connected Overlay, is primarily discussed in studies based on unstructured overlay networks [14]. 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 $t \in T$.

That is, in overlay networks with the satisfaction of 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.

The satisfaction of the relay-free property provides suitability for shortening path length, but it is still difficult to suspend publications 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.

B. 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 of only 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.

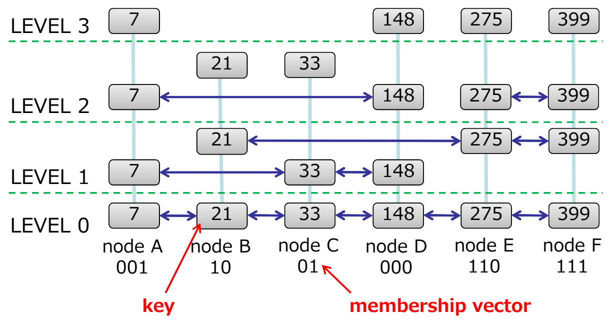


Fig. 4. Example of Skip Graph

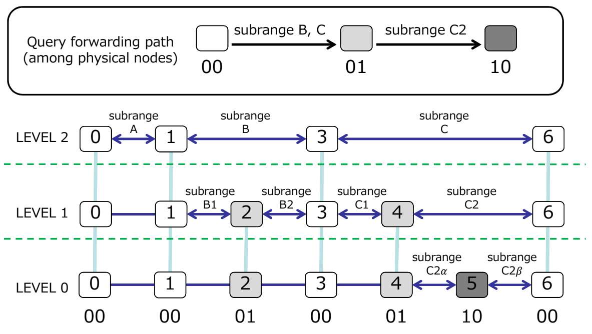


Fig. 5. Query forwarding by multi-range forwarding

IV. Efficient topic-based pub/sub method

As mentioned previously, we have clarified the requirements and formulated them as 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 [15]. Subsequently, we describe the mechanism of suspending publications on the constructed overlay networks.

A. 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 against 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 [17], as shown in Figure 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 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 $\mathcal{O}(\log N)$ of the N participants, while the path length of forwarding queries is also $\mathcal{O}(\log N)$.

B. Multi-key Skip Graph

Multi-key Skip Graph [18] 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 $\mathcal{O}(\log N)$ where N is the number of physical nodes but not virtual nodes.

C. Proposed method

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

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

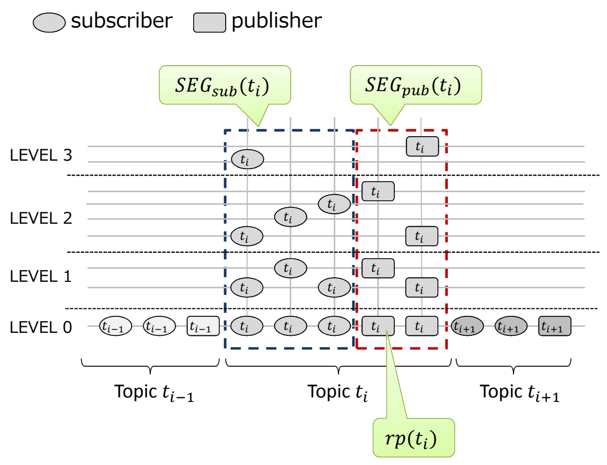


Fig. 6. Ordered relation of nodes in proposed method

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 Figure 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 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 $SEG_{sub}(t)$.

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).

3) Definition of rendezvous point: For $\forall t \in T$, it is ensured that there exists 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 Figure 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.

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. 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 suspend to publishers, by using the range search mechanism with the target range of $SEG_{pub}(t)$.

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.

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. The proposed method adds information about the suspending status onto 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.

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

[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 I
QUALITATIVE COMPARISON OF METHODS

	Suspension	Path length	Storage cost	Responsibility for cost
Proposed	✓	$\mathcal{O}(\log(pub_t + sub_t))$	$\mathcal{O}(\frac{pub_t+sub_t}{N} M \log N)$	Each node
Scribe/Bayeux	✗	$\mathcal{O}(\log N)$	$\mathcal{O}(\frac{pub_t}{N} M + \frac{sub_t}{N} M \log N)$	Multiple nodes
CAN-MC	✗	$\mathcal{O}(d(pub_t + sub_t)^{1/d})$	$\mathcal{O}(\frac{pub_t+sub_t}{N} Md + d)$	Each node

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 $SEG_{pub}(t)$.

D. Qualitative assessment

We give a qualitative assessment of the proposed method in comparison with the methods described in Section II-A. 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 I shows the comparison of the methods. The term "Suspension" in the table denotes the ability to suspend publishing. 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 $\mathcal{O}(\log(pub_t + sub_t))$, 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 $\mathcal{O}(\log N)$. CAN-MC requires $\mathcal{O}(d(pub_t+sub_t)^{1/d})$, 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 $\mathcal{O}(\log N)$ neighbors in the routing table. Thus, the average size of routing tables is $\mathcal{O}(\frac{pub_t+sub_t}{N} M \log N)$. Regarding Scribe/Bayeux, each subscriber forces intermediate nodes on the forwarding path to possess children tables. This storage cost is $\mathcal{O}(\frac{sub_t}{N} M \log N)$ in average. Besides this, each publisher caches the root node of the corresponding topic, then the average cost is $\mathcal{O}(\frac{pub_t}{N} M)$. Accordingly, the total average cost is $\mathcal{O}(\frac{pub_t}{N} M + \frac{sub_t}{N} M \log N)$. With CAN-MC, each publisher or subscriber is inserted onto mini CAN with the cost of $\mathcal{O}(d)$, so the average cost is $\mathcal{O}(\frac{pub_t+sub_t}{N} Md)$. Each node also composes the entire CAN, thus $\mathcal{O}(\frac{pub_t+sub_t}{N} Md + d)$ 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.

"Responsibility for cost" denotes nodes that are responsible for the storage cost. 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 storage cost concerning a subscriber or publisher is taken by multiple nodes which are intermediate nodes on the forwarding path. This seems to be preferable from the viewpoint of load distribution, but it also means that each node cannot predict its load of forwarding. 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 V-C.

V. 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 [19]. Indeed, this type of DHT is implemented by PIAX [20]. Using a DHT on top of Skip Graph is convenient to harmonize the experimental conditions such as the size of routing tables with the proposed method[3]. 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.

A. Number of messages under suspending/publishing

To confirm the ability to suspend publishing, we measured the number of forwarded messages on the overlay

[3]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.

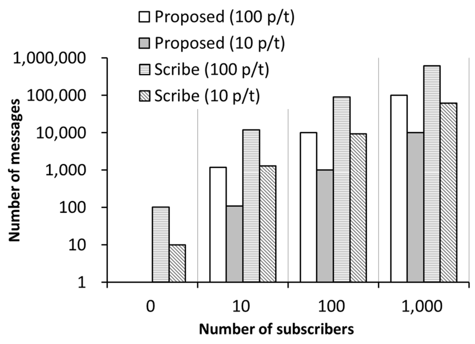


Fig. 7. Number of messages under suspending/publishing

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 7 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.

B. *Length of forwarding path of publications*

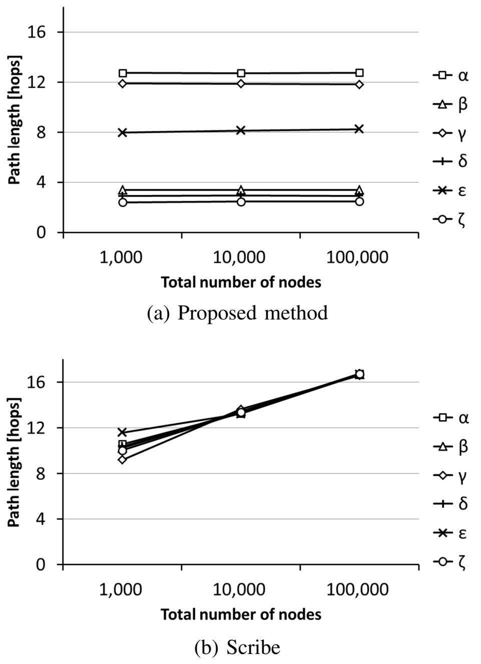
We also evaluated the effectiveness against gratuitous forwarding mentioned in Section II-B. In this experiment, we calculated the average length of paths from each publisher to each subscriber. Here pub_t and sub_t denote the same as described in Section IV-D.

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 II. Each pattern had three different amounts of nodes, $1,000$, $10,000$ and $100,000$. The number of topics in each pattern was keyed to the number of nodes, i.e., the value of "Number of topics" in Table II was 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.

TABLE II
PATTERNS OF THE EXPERIMENT

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



(a) Proposed method

(b) Scribe

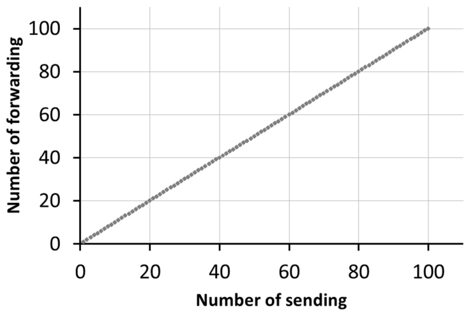
Fig. 8. Length of forwarding path of publications

Figure 8 shows the results. Regarding the proposed method, Figure 8(a) illustrates that the path length was not affected by the total number of nodes but by 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 Figure 8(b) indicate that the path length is not affected by the size of topics, which causes gratuitous forwarding.

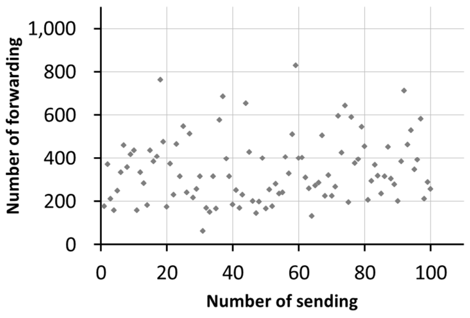
For example, when focusing on patterns of α, γ and ϵ with $100,000$ nodes, the path length is less than 4 hops in the proposed method while Scribe requires more than four times the hops (16 hops).

C. *Correlation between number of sending/receiving and forwarding messages*

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

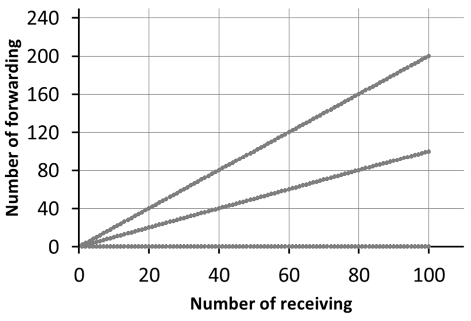


(a) Proposed method

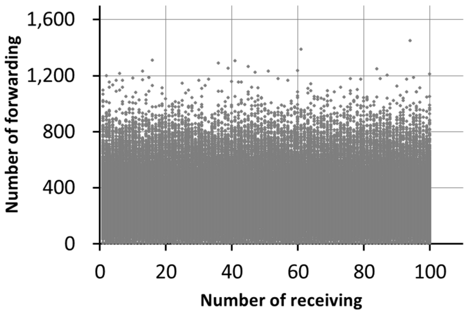


(b) Scribe

Fig. 9. Correlation between number of sending and forwarding



(a) Proposed method



(b) Scribe

Fig. 10. Correlation between number of receiving and forwarding

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 as "Responsibility for cost" in Section IV-D.

If the forwarding load correlates with the load of sending as publishers or receiving 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 sending during the simulation period be $1, 2, ..., 100$.

The simulator constructed overlay networks with the above conditions, and forced publishers to publish at respective intervals. Figure 9 shows the results obtained by plotting all publishers, and Figure 10 shows those by plotting all subscribers. Regarding the proposed method, both the number of sending (Figure 9(a)) and receiving (Figure 10(a)) messages are clearly correlated with the number of forwarding messages. In Figure 10(a), nodes are plotted linearly on three different angled lines. This is because of the characteristic 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[4]. In contrast, the results of Scribe indicate that there are no correlations as shown in Figures 9(b) and 10(b).

[4]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.

TABLE III
CORRELATION COEFFICIENTS AND CONFIDENCE INTERVALS

	Correlation coefficients	99% Confidence intervals
Figure 9(a)	1.0	N/A
Figure 9(b)	0.1310	$-0.1290 \leq \rho \leq 0.3742$
Figure 10(a)	0.5483	$0.5426 \leq \rho \leq 0.5540$
Figure 10(b)	-0.0045	$-0.0126 \leq \rho \leq 0.0037$

The correlation coefficients and their confidence intervals at the 99% level were calculated, as shown in Table III. Note that the confidence interval of Figure 9(a) 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 [21]. Scribe or similar methods receive negative effect from such an imbalance in terms of load predictability. In fact, there is a node that receives a few messages and forwards more than $1,200$ in Figure 10(b).

VI. CONCLUSION

We defined "strong relay-free" as a desirable property of overlay networks for handling exhaust data. Subsequently, we proposed a method of topic-based pub/sub using Skip Graph. The proposed method can construct overlay networks that satisfy the above property. Our method is highly scalable and can suspend publications 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 I. 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.

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

[2] 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.

[3] Oracle, "Java Message Service (JMS)," www.oracle.com/technetwork/java/jms (accessed January 31, 2014).

[4] OASIS, "AMQP," www.amqp.org (accessed January 31, 2014).

[5] 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.

[6] NTT, "Edge computing," www.ntt.co.jp/news2014/1401e/140123a.html (accessed January 31, 2014).

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

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable Content-Addressable Network," *ACM SIGCOMM*, vol. 31, no. 4, pp. 161–172, 2001.

[9] 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, 2004.

[10] 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, 2002.

[11] 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 *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2001, pp. 11–20.

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

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

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

[15] J. Aspnes and G. Shah, "Skip Graphs," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, pp. 37:1–37:25, 2007.

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

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

[18] 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*, vol. 49, no. 9, pp. 3223–3233, 2008 (in Japanese).

[19] N. J. A. Harvey, J. Dunagan, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties," in *USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 9–23.

[20] PIAX, "PIAX: P2P Interactive Agent eXtensions," www.piax.org/en (accessed January 31, 2014).

[21] A. Welhuis, "Twitter and the pareto principle," www.annouckwelhuis.nl/twitter-and-the-pareto-principle-2 (accessed January 31, 2014).