Scalable and Locality-Aware Distributed Topic-based Pub/Sub Messaging for IoT

Yuuichi Teranishi
NICT / Osaka University
Tokyo, Japan / Osaka, Japan
Email: teranisi@nict.go.jp

Ryohei Banno
NTT Network Innovation Laboratories
Tokyo, Japan
Email: banno.ryohei@lab.ntt.co.jp

Toyokazu Akiyama
Kyoto Sangyo University
Kyoto, Japan
Email: akiyama@cc.kyoto-su.ac.jp

***Abstract*—Topic-based pub/sub (TBPS) messaging plays an important role in building event-driven Internet of Things (IoT) applications. In IoT applications, scalability and locality-awareness are important properties that help to achieve low-latency message delivery and efficient usage of network resources. However, none of the existing distributed TBPS methods can simultaneously achieve a sufficient level of both properties. This paper proposes a new TBPS overlay method called 'Skip Graph-based TBPS with Locality-Awareness' (STLA), which extends existing Skip Graph-based TBPS messaging by adding locality-awareness. STLA determines the order of the keys on a Skip Graph overlay network according to the network hierarchy structure using 'locality-aware topic keys' (LATK). Using 'split-forward broadcasting' (SFB) with LATK, the locality-awareness can be dramatically improved. Simulation results show that our method can achieve locality-awareness and reduce the average latency of message delivery for 100,000 subscribers by 76% compared with existing methods. In addition, we have conducted experiments on real distributed data centers using an STLA prototype system, and have confirmed the practicality and feasibility of the proposed method.**

I. Introduction

The Internet of Things (IoT)[1] has received increased attention with the spread of network-connected small devices such as smartphones, sensors, and wearable devices. The IoT is expected to lead to various kinds of smart services. In nature, the smart services are event-driven, i.e., the service delivers notifications to the user's terminal (i.e., smartphone, PC, TV) or actuates certain equipment according to the occurrence of events in the real world. To enable such services, pub/sub messaging[4] is a promising event delivery method that can achieve asynchronous and on-demand information dissemination in the loosely-coupled IoT environment.

As a simple pub/sub method, topic-based pub/sub (TBPS) is useful for event notifications. TBPS protocols such as MQTT[2] and AMQP[3] are already widely utilized by many IoT applications. In TBPS, a message is published to a channel, which is known as a 'topic.' Users may subscribe to multiple topics, and all users subscribed to a topic receive every message published to that topic. Publishers specify a predefined topic to each message, i.e., "river water-level at point 1," "video camera in room 2," and so on, instead of referencing the network address of each specific receiver. The pub/sub system then delivers the message to all subscribers that have requested messages on that topic.

TBPS systems for IoT applications must have scalability. When an event impacts people over a wide area or becomes popular, a huge number of users may subscribe to the topic related to that event. For example, the topic of river water-level observed at an upper stream may attract interest from people who live downstream. When a well-known open event, e.g., a live performance or flash mob, occurs at some open-air stage or street, a massive number of people may subscribe to topics such as "video data stream captured at the place." The event delivery system must be able to tolerate such situations. That is, the event delivery performance must be maintained, even if there are a huge number of subscribers.

Another important property of TBPS systems for IoT applications is locality-awareness. Publishers and subscribers of IoT applications are often located geographically close to each other, because events that correspond to certain sensor data often correspond to a location. For example, people who live near a place where heavy rain is falling have a stronger interest about this phenomenon than those living farther away, as the phenomenon is more likely to affect them. Generally, publishers/subscribers located geographically close to each other are accommodated on the same local network or local networks that are topologically adjacent. From the network point of view, if the publisher and subscriber are accommodated in the same or neighboring local network, the traffic between them should not affect other parts of the wide network. The volume of traffic can be a serious problem when large amounts of continuous data are published by sensors in the IoT applications. In addition, if the messaging route is limited to the local network or neighboring local networks, the latency decreases, which is desirable for IoT applications that require real-time responses.

The traditional pub/sub systems, which include existing TBPS implementations, have a centralized broker server for managing topics, publishers and subscribers. The centralized broker gathers all topics and arranges message forwarding between publishers and subscribers. However, such centralized TBPS systems suffer from poor scalability[5]. In addition, they have little locality-awareness, because the centralized broker has to manage all subscribers, and the location of this centralized broker cannot be changed.

To improve the scalability of TBPS messaging, many approaches with no centralized broker have been proposed. These approaches treat distributed brokers as peers in a peer-to-peer system, and construct an overlay network. To enable a distributed approach to achieve scalability, structured overlay-based multicasting is a practical choice. There

are some existing studies to achieve pub/sub overlay, such as distributed hash table (DHT)-based approaches[6], [7], [8], overlay networks with quorum-based topology[9], hybrid overlay approaches[10], [11], and Skip Graph[12]-based approaches[13].

However, none of these existing studies has given sufficient consideration to network locality on the structured overlay. The message delivery path generated by a structured overlay has a mismatch with respect to the physical network topology. In this case, inefficient and redundant message exchanges arise, especially when there are a huge number of subscribers. For example, a message from a publisher may be forwarded via other networks to a subscriber, even when the publisher and the subscriber are on the same network.

To solve such problems, we propose a novel and practical TBPS messaging method called 'Skip Graph-based TBPS with Locality-Awareness' (STLA). STLA can achieve high scalability and locality-awareness, which, to the best of the authors' knowledge, none of the existing distributed TBPS methods can achieve.

II. Distributed TBPS by Overlay

A. Assumed TBPS Architecture

To enable a distributed approach to achieve scalability and locality-awareness, we focus on an architecture using small, distributed brokers, as shown in Fig. 1. Each broker corresponds to the network entity located physically close to each end device. The end devices correspond to IoT entities such as sensors, smart-phones, and appliances.

The broker can run on any networked computer that is located close to the end device. The computer may be a server at the distributed data center, a Wi-Fi access point with computing power, a personal desktop computer, and so on. The model is compatible with hierarchical cloud architectures recently proposed for IoT (e.g., the so-called 'fog computing' [14] architecture).

A broker contains a subscriber or a publisher of a topic if one of the devices accommodated on the broker attempts to subscribe/publish to the topic. The brokers then construct an overlay network for message delivery on each topic. Each broker joins the overlay network to handle distributed message deliveries. The end device can be a broker if it is able to handle the overlay function. Hereafter, an entity that joins the overlay is denoted as a 'node.' The node can be a publisher, a subscriber, or both. Brokers can contain multiple publishers or subscribers across different topics.

B. Related Works

Scribe[6] and Bayeux[8] are DHT-based algorithms for distributed TBPS. In such approaches, a spanning tree is built for each topic, with a rendezvous node delivering messages to those nodes that have subscribed to the topic. However, this approach forces nodes to process messages that they have not subscribed to, simply because they happen to be on the path towards the rendezvous node. Because the nodes join the DHT network using a hash function, those on the delivery path are selected randomly. Consequently, the nodes on the delivery path can be located on multiple physical networks.

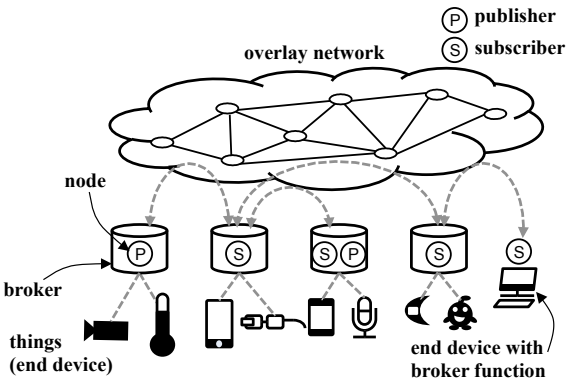

Fig. 1: Distributed TBPS system by overlay network

In addition, such systems suffer from a huge amount of wasted network overhead. Even when there are no subscribers, messages must be delivered to the rendezvous node. This property is unfavorable in IoT applications that generate huge amounts of periodic sensor data that is of limited interest.

Vitis[10] builds a hybrid overlay of structured and unstructured overlays for TBPS. It constructs gossip-based unstructured overlays for groups of nodes that are subscribed to the same or similar topics. These groups of nodes are connected with each other via the structured overlay similar to the DHT-based algorithms. There is an extension of Vitis that treats locality-awareness[11]. This method selects neighboring nodes which have smaller round-trip latency to form the group. However, as the number of subscribers grows, Vitis encounters the same problem as DHT-based algorithms. In Vitis, the number of nodes in the same group is limited, which means that the structured overlay needs to treat a larger number of groups as the number of subscribers grows.

The article [13] proposed a Skip Graph-based TBPS (hereafter, ST) method. ST partially solves the problems found in previous methods, and provides 'strong relay-free' topology on the Skip Graph. In this structure, publishers and subscribers of the same topic are located adjacently on the Skip Graph (further details are given in Section III-B). Thus, in ST, publishers can detect the absence of a subscriber through neighbor notifications, thus avoiding the waste of network resources. As long as the publisher and the subscriber are contained in the same broker, locality-awareness is also achieved. However, when subscribers are contained in multiple different brokers, locality-awareness is lost. Under ST, the message delivery path encounters a mismatch with the physical network topology.

To achieve sufficient locality-awareness, we propose a TBPS messaging method named *Skip Graph-based TBPS with Locality-Awareness* (STLA), which extends the ST method.

III. TBPS by Skip Graph

In this section, we present an overview of the technologies behind our proposal, namely the Skip Graph algorithm and the existing ST method, and state some theorems.

A. Skip Graph

Skip Graph is an algorithm for structured overlay networks that provide a scalable node search function. Each node in

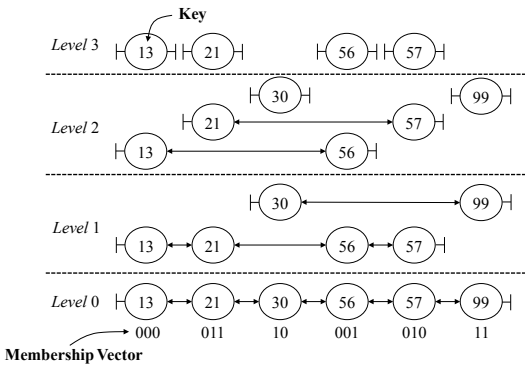

Fig. 2: An example of Skip Graph

a Skip Graph overlay network has two fields: a key and a membership vector.

Let $m(x)$ denote the membership vector of node x. The elements of $m(x)$ belong to a finite alphabet set $\sum$. The membership vector has an average length of $O(\log N)$ where N indicates the number of nodes.

Fig. 2 shows an example Skip Graph structure. There are several levels that are determined by the membership vector. Level 0 is a doubly linked list that consists of all nodes sorted in order of their keys.

When denoting the l-length prefix of $m(x)$ as $m(x) \restriction l$, two nodes x and y belong to the same list at level i iff $m(x) \restriction i = m(y) \restriction i$.

This property enables Skip Graph to have long-distance skip links. 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 a long distance at the higher levels and gradually moves down to level 0.

The following theorem was proved in the paper that introduced Skip Graph[12].

Theorem 1. *The expected number of hops required to deliver a message with N nodes on Skip Graph is* $\log N$.

Normally, the alphabet size of the Skip Graph $|\sum|$ is set to 2. Therefore, it is hereafter assumed that $|\sum| = 2$.

B. ST: Skip Graph-based TBPS

In ST, each subscriber or publisher joins a Skip Graph network based on its key, which includes a string for the topic name.

One node can have multiple keys containing multiple publishers or subscribers of different topics. To distinguish subscribers with the same topic name on the Skip Graph network, a unique suffix is added to the topic name on each key. There are a subscriber segment and a publisher segment on each topic. Fig. 3 shows an example of the ST overlay structure. This structure enables publishers to detect the existence of subscribers using the neighbor link at level 0, and thus redundant traffic from publishers is reduced when there are no subscribers.

Under the ST method, the following theorem holds, as it uses Skip Graph as the base structure.

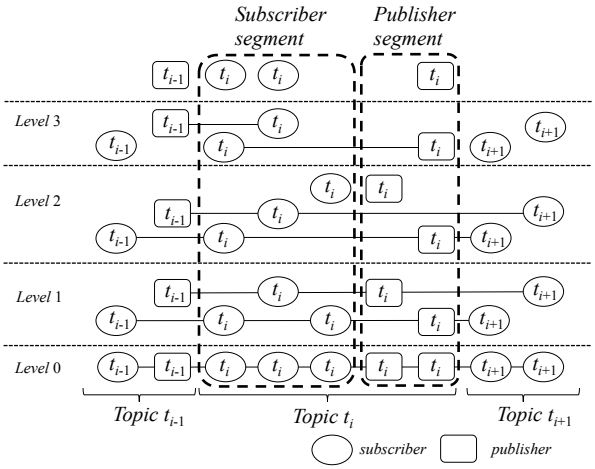

Fig. 3: An example of Skip Graph-based TBPS

Theorem 2. *The expected number of hops required to deliver a message for a topic in ST is* $\log M$*, where M is the number of publishers and subscribers of the topic.*

IV. STLA: Skip Graph-based TBPS with Locality-Awareness

This section describes our proposal TBPS method STLA. We first introduce the hierarchical network model used in STLA. To combine hierarchical properties with ST, we use 'locality-aware topic keys' (LATK) for the Skip Graph keys. On the overlay structure, a multicast method called 'split-forward broadcasting' (SFB) is used to achieve locality-awareness in STLA. The following subsections explain these features and the process of STLA.

A. Hierarchical Network Model

STLA uses a hierarchical network model to represent approximate network locations. The hierarchical network model is similar to the model introduced by Shao et al. [15].

The Internet has natural hierarchical properties such as the access networks to which terminals connect, the Internet service providers' networks to which they contract, and the wide-area networks between countries and continents. The data center network also has hierarchical properties, such as the layer 2 network under the top-of-rack switch, the layer 3 local area network within a data center, and the wide-area network among data centers.

The physical network model used in the proposed system is the hierarchical structure shown in Fig. 4. We call each hierarchical network a *cluster*. Each cluster in a hierarchical layer accommodates multiple clusters in the lower layer. The clusters in the bottom layer of the hierarchy accommodate the nodes. We denote the set of clusters in layer i as L_i. In L_0, the top layer in the hierarchy, there is only one cluster, which contains all of the clusters in the world.

Each cluster has a 'cluster identifier.' The cluster identifier is denoted as $C_{l,q}$, where l is the layer of the hierarchy and q is the unique identifier in the cluster of layer l. The unique identifier must be a comparable value, which means

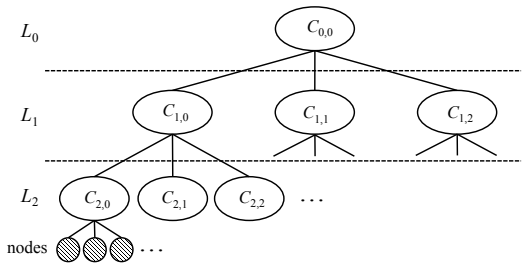

Fig. 4: Network hierarchy model

that comparative operators such as $=, <, >, \geq, \leq$ must be applicable to any two unique identifiers. For example, this can be satisfied by adding a sequence number to each cluster. Each cluster has a fully-qualified cluster identifier (FQCI), which is an ordered list of cluster identifiers. For instance, the FQCI of the cluster that accommodates the hatched nodes in Fig. 4 is denoted as $\{C_{0,0}, C_{1,0}, C_{2,0}\}$.

We assume that each node knows its FQCI in advance. This hierarchical property of a node can be estimated by the domain name, global IP address, and so on. It is also possible to extract the physical network structure using a protocol such as ALTO[16]. This assumption is increasingly practical, because widespread software-defined network platforms can provide such information to the applications[17].

In this physical network model, we define the requirements of locality-awareness as follows:

- Minimize the inter-cluster hops
  To achieve locality-awareness, the number of inter-cluster hops should be minimized in the message delivery. Reducing the inter-cluster hops required for message delivery also reduces the message delivery latency.
- Minimize the total inter-cluster traffic
  To maintain the locality-awareness and run the system efficiently, the message exchange between clusters should be minimized. By reducing the message exchange between clusters, redundant network resource usages are also reduced.

If scalability is not considered, the first requirement can be satisfied easily. The optimal case is when the publisher sends all messages directly to the subscribers. However, in this case, inter-cluster traffic becomes large, because every subscriber in different clusters requires inter-cluster message delivery. To realize scalability using the structured overlay network-based TBPS of previous methods, these properties cannot be controlled, as the multicast delivery path is determined at random.

The basic idea of STLA is to construct a structured overlay on the basis of the hierarchical clusters for each topic, and generate a loop-free multicast tree among the clusters.

B. Locality-Aware Topic Keys

Fig. 5 shows the structure of LATK. In this figure, the structure of LATK is compared with that of the keys used by ST.

The 'topic' field corresponds to the topic itself, and can be an arbitrary string. Thus, the topic is comparable via dictionary order. The 'type' field contains a bit to show whether the node holding the key is a publisher or a subscriber. A bit value of 0 denotes a subscriber, and 1 denotes a publisher. The 'node ID' field is the unique suffix of ST.

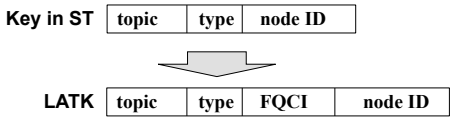

Fig. 5: Format of LATK

LATK adds the FQCI before the 'node ID' field. Thus, the keys of the same topic are ordered by the FQCI. We define the order of FQCI as lexicographical order using comparator $<_{\text{FQCI}}$ for two FQCIs, namely

$$\{C_{0,a_0}, C_{1,a_1}, \ldots, C_{H-1,a_{H-1}}\} <_{\text{FQCI}} \{C_{0,b_0}, C_{1,b_1}, \ldots, C_{H-1,b_{H-1}}\} \iff a_0 < b_0 \vee \{(\exists m > 0)(\forall i < m)(a_i = b_i) \wedge (a_m < b_m)\}$$

where H corresponds to the height of the cluster hierarchy.

Thus, the LATK itself is also a comparable value. Hereafter, the comparator $<$ of LATK is denoted as $<_{\text{LATK}}$. A LATK for topic t with type p (0 or 1) for node x is denoted as $LATK(t, p, x)$. In STLA, to subscribe to a topic, each node simply joins a Skip Graph network using LATK.

C. Multicasting on Skip Graph Structure

For multicast routing on Skip Graph, we propose the use of SFB[18]. The original Skip Graph paper proposed a broadcasting method over a certain range, but this generates a lot of duplicate received messages. SFB can achieve the same hop number with a broadcasting scheme that incurs no duplicate message deliveries. SFB performs message forwarding across a specific range as follows:

Suppose a node $u(u.key = y)$ received a message that has the range $[x, y]$. Let v_i represent u's left neighbor in level i. If $x \leq v_i.key$, then u forwards the message to v_i with the split range $[x, v_i.key]$. The forwarding process is performed for each level of the node u, from higher levels to lower levels. In the level $i-1$, the rest of the split range $(v_i.key, y]$ is processed. Upon receiving a query, each node performs the same process, until every node holding the key has received the message. This process runs simultaneously for the right neighbors in each level.

D. Message delivery sequence

Fig. 6 shows the message delivery sequence under STLA. For a better understanding, the Skip Graph network in this figure has an ideally balanced structure. The order of a subscriber segment and a publisher segment of a topic is the same as for ST. Thus, STLA can detect the absence of subscribers in the same way as ST.

To maintain locality-awareness between the publisher and subscribers, the publisher starts multicasting from a node that belongs to the same or the closest cluster. We call this the 'start node.' To select the start node, the publisher of topic t searches for a subscriber node with key K_t, which satisfies following:

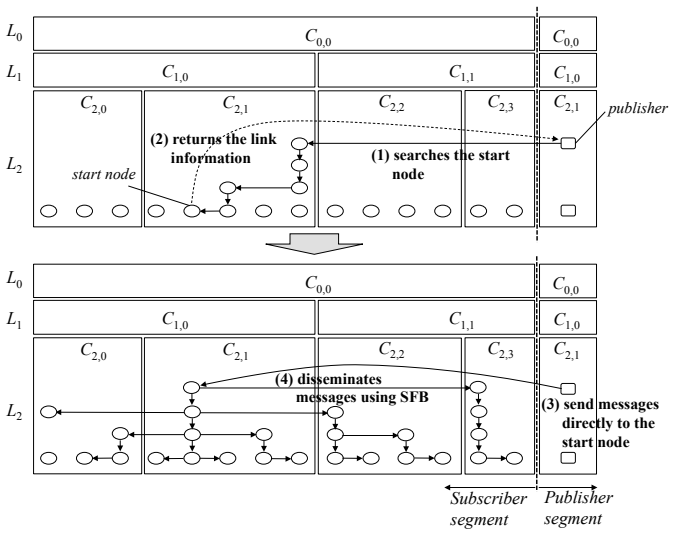

Fig. 6: Message delivery sequence

$$K_t = \arg\max_{n \in M} \ LATK(t, 0, n)$$
$$where\ M = \{s | LATK(t, 0, s) <_{\text{LATK}} LATK(t, 0, p)\}$$

where p is the publisher node using Skip Graph ((1) in the figure). That is, the publisher searches for the largest key, that is not greater than the LATK of the publisher node as a subscriber. If there is one or more subscribers belonging to the same cluster with the publisher, the node in the same cluster with the publisher matches as K_t. Otherwise, a node in the neighbor cluster with the publisher matches as K_t. To ensure to select a node on the closest cluster as a start node, the node matched as K_t returns the link information of the node K_t itself and the adjacent node at the right side of K_t on level 0 of Skip Graph (2).

The link information includes network address and LATK of the node. The publisher selects the start node that belongs to the closest cluster from the obtained link information. Then the publisher sends messages directly to the start node without using the Skip Graph structure (3). On receiving the message, the start node disseminates the message to the subscribers using SFB (4). By using SFB, each delivery path to the subscribers never passes the same cluster twice. Thus a loop-free multicast tree is generated among the clusters.

E. Analysis

In this subsection, we analyze the performance of STLA. As we have described, inter-cluster hops affect the message delivery performance. In STLA, the following theorem exists for inter-cluster hops.

Theorem 3. *The upper bound of the average inter-cluster hop count at layer i in STLA is*

$$\begin{cases} |L_i| & (|L_i| < \log M) \\ \log M & (|L_i| \geq \log M) \end{cases}$$

where M is the number of subscribers for the topic.

Proof. The message never passes through the same cluster twice at each layer under message forwarding in SFB. Therefore, if $|L_i| < \log M$, the message passes, at most, $|L_i|$ clusters. Otherwise ($|L_i| \geq \log M$), since every hop passes a different cluster in the worst case, the upper bound of the average inter-cluster hop count becomes $\log M$ by Theorems 1 and 2. □

In STLA, the 'size of cut' (the number of links exist between adjacent two nodes) of the message delivery tree has the following property.

Lemma 1. *The expected total size of cut in the message delivery tree structure of SFB is $\frac{M \log M}{2}$, where M is the number of subscribers.*

Proof. In a Skip Graph network, if M is doubled, the size of cut of the message delivery tree of a topic also expected to be doubled. In addition, the size of cut is expected to be increased half of the number of nodes, because a link is added that crosses half of all the nodes, on average. Therefore, the expected size of the cut in a Skip Graph network with M nodes is $\sum_{i=1}^{\log M} (\frac{2^i \times 2^{\log M - i}}{2}) = \frac{M \log M}{2}$ □

Using this property, we can derive a theorem that states the expected total traffic between clusters for message delivery in STLA.

Theorem 4. *In a uniform distribution, the expected total inter-cluster traffic at layer i in STLA is $\frac{|L_i|}{2} \log M$.*

Proof. A cluster boundary exists between neighbor nodes at layer i with probability $\frac{|L_i|}{M-1}$ in the uniform distribution. Since the expected total size of cut is $\frac{M \log M}{2}$ (Lemma1), the expected total inter-cluster traffic at layer i is $\frac{|L_i| M \log M}{2(M-1)} \approx \frac{|L_i|}{2} \log M$ when M is sufficiently large. □

In the higher layers of the cluster hierarchy, the number of clusters is smaller but the overhead between clusters becomes large, which affects the overall delivery performance. Therefore, the overhead becomes closer to $O(\log M)$, when $|L_i|$ is smaller. The uniform distribution is assumed in the above theorem because it is the worst case for the total inter-cluster traffic.

V. Performance Evaluation

In order to show the effectiveness of SLTA, we evaluated its performance through simulations and experiments. ST is used as a comparative method. In the evaluation, ST uses SFB for multicasting for evaluations on the same precondition. The nodes are distributed according to the Zipf distribution[19] and uniform distribution. The Zipf distribution can reproduce the scenarios in which the popularity of subscribers is concentrated within a small number of clusters.

A. Simulation Results

To evaluate scalability, we assumed there was one publisher and varied the number of subscribers from 10 to 100,000. We evaluated two scenarios, an Internet scenario and a data center (DC) scenario. In the Internet scenario, Layer 0 is the world containing all nodes, Layer 1 is the continents layer, and

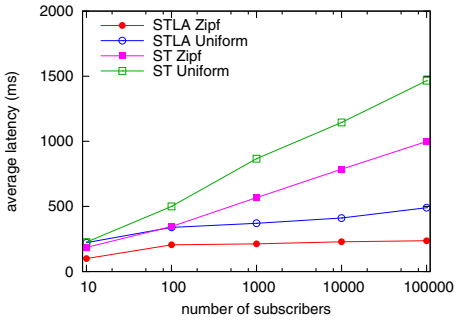

Fig. 7: Average latency by changing # of subscribers (Internet scenario)

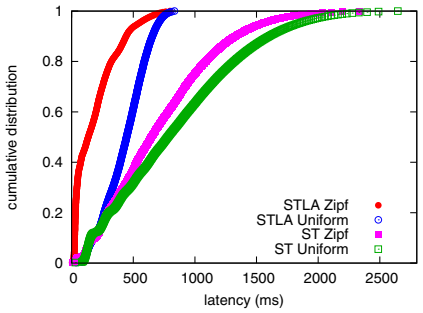

Fig. 8: Cumulative distribution of latency for 100,000 nodes (Internet scenario)

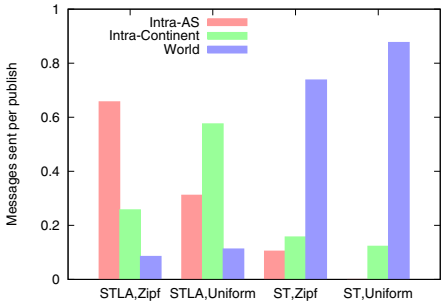

Fig. 9: Inter-cluster traffic for 100,000 nodes (Internet scenario)

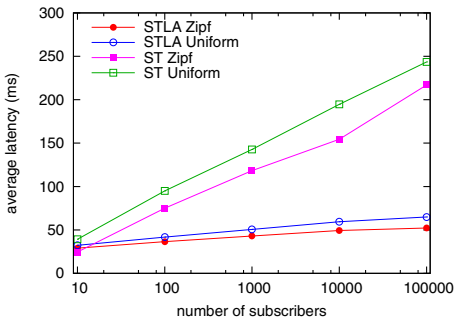

Fig. 10: Average latency by changing # of subscribers (DC scenario)

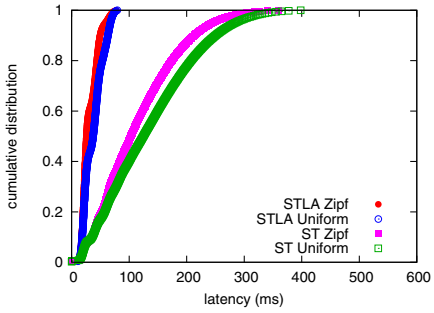

Fig. 11: Cumulative distribution of latency for 100,000 nodes (DC scenario)

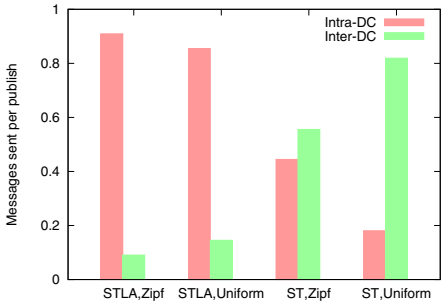

Fig. 12: Inter-cluster traffic for 100,000 nodes (DC scenario)

TABLE I: Simulation parameters

Parameters	Internet scenario	DC scenario
Hierarchy height	3	2
# of clusters	1, 7, 400	1, 5
Latency of each cluster(ms)	100, 20, 2	20, 0.2
Distribution	Uniform, Zipf	Uniform, Zipf

Layer 2 is the autonomous systems (ASes) layer. We suppose 7 continents, with 400 ASes in each continent. The latency between continents was set to 100ms, that between ASes in the same continent was set to 20ms, and the intra-AS latency was set to 2ms. These are the same settings as used in [15].

In the DC scenario, Layer 0 is the inter-DC layer and Layer 1 is the DC layer. We assumed there were five data centers, and set the latency between DCs to 20 ms, whereas the latency inside DCs was set to 0.2 ms. These values are taken from measurements of an actual data center network in the JOSE test-bed[21]. In both scenarios, the latency varied according to the Gaussian distribution. Table I presents a summary of the parameters in both scenarios.

The evaluation results are shown in Figs. 7-12. Figs. 7-9 give the results for the Internet scenario, and Figs. 10-12 present the results for the DC scenario.

Figs. 7 and 10 show the average latency of message deliveries on a topic for different numbers of nodes and popularity in each scenario. As shown in the figures, ST and STLA both maintain scalability as the number of subscribers grows. STLA exhibits a drastic improvement in latency, especially when there are a large number of subscribers. The performance of STLA was improved better when the popularity follows the Zipf distribution. For 100,000 subscribers, the latency was reduced by 76% in both the Internet and DC scenarios.

Figs. 8 and 11 show the cumulative distribution of the latency when there are 100,000 nodes in each scenario. We can see that STLA not only reduces the average latency, but also reduces the latency in the worst case. The difference in latency between ST and STLA grows with the number of delivery nodes. Under ST, the Internet scenario with a Zipf distribution encounters a maximum latency of almost 2300 ms. However, under STLA, the worst-case latency is about 800 ms. Note that the simulation results do not include the process overheads and the delivery latency between the broker and the end device.

Figs. 9 and 12 show the routing locality of message deliveries on a topic in each scenario. Routing locality is represented as the ratio of messages sent per publisher in each layer of the hierarchical structure. From the figures, we can confirm that STLA achieves higher routing locality than ST. The Zipf distribution results in higher routing locality than the uniform distribution. That is because the Zipf distribution causes most of the nodes to belong to a few clusters, which makes it easy to increase the routing locality.

B. Experimental Results

To confirm the practicality and feasibility of STLA, we conducted experiments in a real data center environment. This subsection presents latency measurements obtained from a

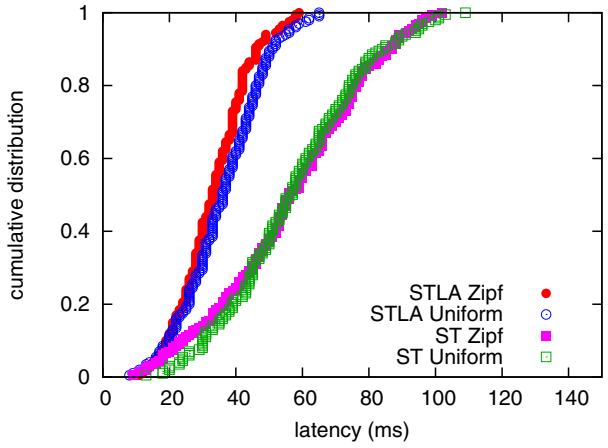

Fig. 13: Cumulative distribution of latency at the experiments by prototype implementation using 200 distributed nodes.

prototype implementation. The STLA was implemented on open source peer-to-peer middleware called PIAX[20]. The User Datagram Protocol (UDP) was used for the network layer protocol. Brokers made by STLA were deployed at three sites on the JOSE test-bed[21], namely at Yokosuka, Kyoto, and Kanazawa in Japan. In the evaluation, we considered the case in which 200 subscribers were subscribed to one topic.

As in the simulations, we applied both uniform and Zipf subscriber distributions. In the uniform distribution, 66 or 67 subscribers were located at each data center. For the Zipf distribution, we placed 102 subscribers in Kanazawa, 64 subscribers in Yokosuka, and 34 subscribers in Kyoto. Each broker was run on one virtual machine and contained one subscriber. A publisher in the Kanazawa data center published messages. We measured the message delivery latency among brokers.

Fig. 13 shows the cumulative distribution of the measured latency. The minimum value of 100 message deliveries on a topic is plotted. With the Zipf distribution, the delivery completion latency was 59ms under STLA and 102ms for ST, whereas for the uniform distribution, STLA averaged 65ms against 109ms for ST. That is, the delivery completion time was improved by 41-43% under the proposed method.

This experiment shows that the latency exhibits a similar tendency to that of the simulations of DC scenarios, although there are additional process overheads.

VI. Conclusion

In this paper, we have presented a practical distributed TBPS method for IoT applications called STLA, which satisfies both scalability and locality-awareness requirements.

In the proposed method, LATKs were used to reflect the physical network topology on the overlay's logical network structure. Using the SFB multicast method with LATK dramatically improved locality-awareness.

Simulations confirmed that STLA could achieve locality-awareness and reduced latency in both Internet and DC network models. We confirmed that the STLA overlay with 100,000 nodes reduced the latency by 76% compared with existing methods. Experimental evaluations using a prototype implementation demonstrated the practicality and feasibility of STLA.

We believe that STLA can be a practical and valuable message exchange foundation for IoT applications, especially in evolving hierarchical cloud architectures.

In future work, we will consider reducing the redundant traffic between clusters to save network and computation resources. In addition, an aspect worth considering is the impact of mobility and network topology changes for end devices. Implementations compatible with existing TBPS protocols such as MQTT and AMQP will also be our future work.

References

[1] S. Hodges, et al. Prototyping Connected Devices for the Internet of Things. *IEEE Computer*, pp. 26 - 34, 2013.
[2] MQTT Version 3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf (accessed Mar. 31, 2015).
[3] Advanced Message Queuing Protocol. http://www.amqp.org (accessed Mar. 31, 2015).
[4] P.T. Eugster, et al. The Many Faces of Publish / Subscribe. *ACM Computing Surveys*, vol. 35, no. 2, pp. 114 - 131, 2003.
[5] K. Paridel, et al. Middleware for the Internet of Things, Design Goals and Challenges. *Proc. of CAMPUS 2010, EC EASST*, 2010.
[6] M. Castro, et al. SCRIBE: A Large-scale and Decentralized Application-Level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, 2002.
[7] S. Ratnasamy, at al. Application-Level Multicast using Content-Addressable Networks. *Proc. International COST264 Workshop on Networked Group Communication*, 2001.
[8] S.Q. Zhuang, et al. Bayeux : An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2001.
[9] Y. Sun, et al. A Low-Delay, Lightweight Publish/Subscribe Architecture for Delay-sensitive IOT Services. *Proc. of IEEE International Conference on Web Services*, 2013.
[10] F. Rahimian, et al. Vitis: A Gossip-based Hybrid Overlay for Internet-scale Publish/Subscribe Enabling Rendezvous Routing in Unstructured Overlay Networks. *Proc. of IEEE IPDPS 2011*, 2011.
[11] F. Rahimian, et al. Locality-Awareness in a Peer-to-Peer Publish/Subscribe Network. *Lecture Notes in Computer Science*, Vol. 7272, 2012.
[12] J. Aspnes, G. Shah. Skip Graphs. *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, 2007.
[13] R. Banno, et al. Designing Overlay Networks for Handling Exhaust Data in a Distributed Topic-based Pub/Sub Architecture. *Journal of Information Processing*, Vol.23, No.2, 2015.
[14] F. Bonomi, et al. Fog Computing and Its Role in the Internet of Things. *Proc. of MCC 2012*, 2012.
[15] X. Shao, et al. A Low Cost Hierarchy-Awareness Extension of Skip Graph for World-Wide Range Retrievals. *Proc. of IEEE COMPSAC 2014*, 2014.
[16] Application-Layer Traffic Optimization. http://datatracker.ietf.org/wg/alto/charter/ (accessed Mar. 31, 2015).
[17] T. Akiyama, et al. SAPS: Software Defined Network Aware Pub/Sub. *Proc. of IEEE COMPSAC 2015*, 2015.
[18] R. Banno, et al. SFB: A Scalable Method for Handling Range Queries on Skip Graphs. *IEICE Communications Express*, Vol. 4, No. 1, 2015.
[19] G. Zipf. Human Behaviour and the Principle of Least-Effort. *Addison-Wesley*, 1949.
[20] PIAX. http://www.piax.org/ (accessed Mar. 31, 2015).
[21] JOSE. http://www.nict.go.jp/en/nrh/nwgn/jose.html (accessed Mar. 31, 2015).