Journal of Information Processing Vol.24 No.4 635–646 (July 2016)

[DOI: 10.2197/ipsjjip.24.635]

Regular Paper

Scalable Pub/Sub System Using OpenFlow Control

Toyokazu Akiyama[1,a)] Yuuichi Teranishi[2] Ryohei Banno[3]
Katsuyoshi Iida[3] Yukiko Kawai[1]

Received: October 20, 2015, Accepted: April 5, 2016

Abstract: Pub/Sub communication model becomes a basis of various applications, e.g., IoT/M2M, SNS. These application domains require new properties of the Pub/Sub infrastructure, for example, supporting a large number of devices in a widely distributed manner. In order to meet the demands, we proposed Scalable Pub/Sub System using OpenFlow Controller, which we call SDN Aware Pub/Sub (SAPS). SAPS utilizes the both Application Layer Multicast (ALM) and OpenFlow based multicast (OFM). A simulation was done for evaluating the hybrid architecture in traffic and transmission delay reduction. The result shows that in the tree topology, even if it has only 100 subscribers, OFM can reduce inter-cluster traffic 71.6% with 16 clusters compared to ALM-LA. It can also reduce maximum inter-cluster hops 87.5%. On the other hand, with only 100 subscribers, almost all of switches are involved in the OFM tree construction and consume flow table space for the topic. It indicates that our hybrid approach is effective in Pub/Sub optimization considering the resource limitation of OpenFlow switches.

Keywords: Pub/Sub, Application Layer Multicast, OpenFlow multicast, Software Defined Networking

1. Introduction

The Publish/Subscribe (Pub/Sub) communication model becomes a basis of various applications, for example, IoT/M2M, SNS and so on. Recent applications require massive amount of messaging among the end entities. Especially in IoT applications like Smart City, huge amounts of widely distributed sensors and actuators require interactions each other [1]. Constructing a new messaging infrastructure which can localize the inter-entity communication and reduce the core traffic is an emerging requirement.

In many Pub/Sub systems, subscribers register subscriptions to an intermediary broker and publishers post messages to the broker. In order to support geographical and load scalability, it requires multiple brokers to be deployed in a distributed manner. When the number of broker increases, the *multicast* becomes desirable for inter-broker communications instead of the broadcast. Application Layer Multicast (ALM) can be applied for inter-domain services on behalf of IP multicast [2]. ALM can reduce the load of source node and unnecessary message delivery to the brokers without subscribers. On the other hand, since ALM cannot reflect physical structure to the logical message delivery path, it may cause unnecessary message delivery. There are mainly two ways to solve this problem. One is to solve it by optimizing ALM using under layer information [3], [4], the other is to optimize under layer forwarding by directly controlling network switches. The former approach is an easier way in practical environment because of the difficulty of cross layer control requirements. However, recently, newly developed technology, Software Defined Network (SDN), makes an evolution in this field. It provides an application programming interface to enable more interactive control with higher layer mechanisms. And also, there are several research developing new multicast mechanism utilizing SDN functionalities [5], [6].

In this paper, we focus on developing the latter approach by applying SDN technology. We propose a new Pub/Sub infrastructure which uses application layer multicast in conjunction with lower layer multicast for establishing geographical and load scalability as well as recognition of application layer demands in a lower layer.

The rest of the paper is organized as follows. In Section 2, we will discuss related work to describe the target issue of this paper and then show you the details of the proposed Pub/Sub infrastructure in Section 3. In Section 4, we will show evaluation results and discuss the issues in the current design of the proposed hybrid architecture. Finally, we will conclude this paper in Section 5.

2. Related Works

In this section, we will discuss related work to clarify the target issues to be solved by our proposal. **Figure 1** describes issues of Pub/Sub system and target of our study using an example Pub/Sub system deployment for an IoT application. In this deployment, sensors and actuators are connected to a broker running on a home gateway instrument and home gateways are also connected to a broker at a Regional Data Center (RDC) (Fig. 1 (1)). While direct communication among brokers on home gateways can reduce resource consumption of nodes at RDC, in this example, each broker on home gateway is assumed to communicate only with a RDC broker from security reasons. The following discussion can also be applied to the case with such end-to-end direct communication if security policy permits. The broker at a RDC

[1] Kyoto Sangyo University, Kyoto 603–8555, Japan
[2] National Institute of Communication Technology, Koganei, Tokyo 184–8795, Japan
[3] Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[a)] akiyama@cc.kyoto-su.ac.jp

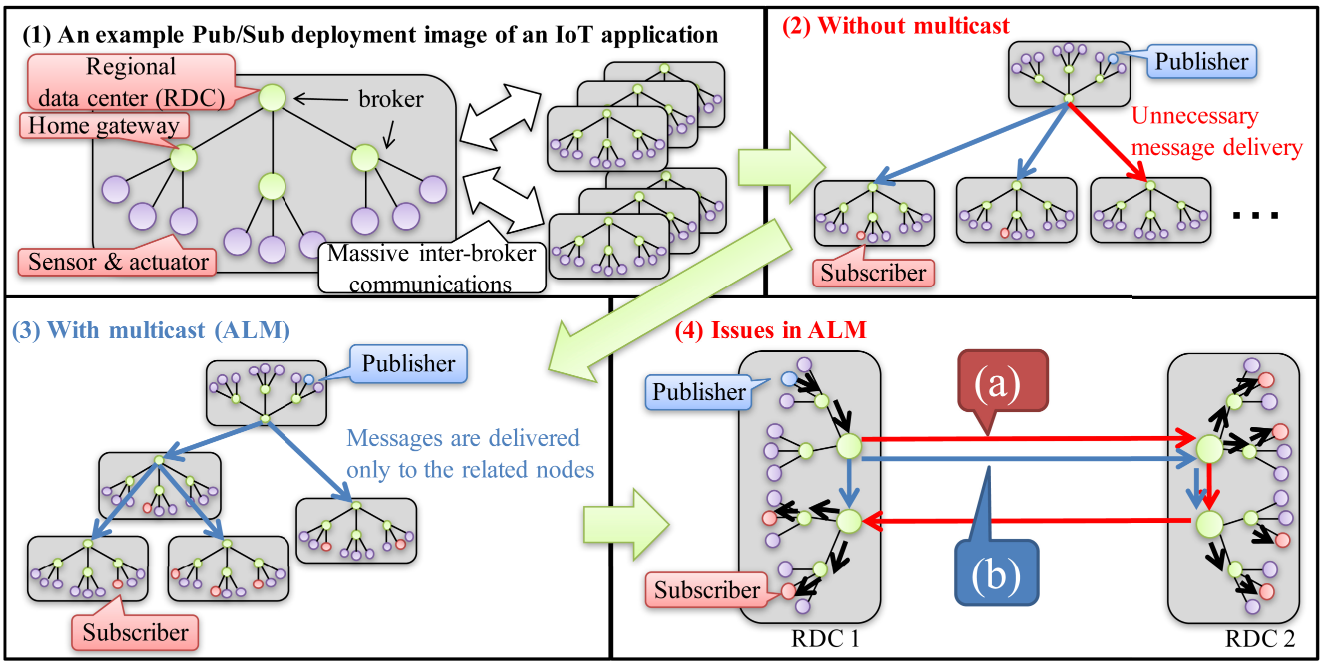

(1) An example Pub/Sub deployment image of an IoT application
Regional data center (RDC)
broker
Home gateway
Sensor & actuator
Massive inter-broker communications
(2) Without multicast
Publisher
Unnecessary message delivery
Subscriber
(3) With multicast (ALM)
Publisher
Messages are delivered only to the related nodes
Subscriber
(4) Issues in ALM
Publisher
(a)
(b)
Subscriber
RDC 1
RDC 2

Fig. 1 Issues of Pub/Sub system and target of our study.

must communicate with other brokers at the same or the other RDC. Here, sensors and actuators become a publisher or a subscriber of Pub/Sub system and their messages are aggregated to massive amount of inter-broker traffic. If a simple flooding is applied for the inter-broker communication (Fig. 1 (2)), unnecessary message delivery will occur and the load of sender node becomes large. ALM can remove such unnecessary message delivery by maintaining logical links to the related nodes (Fig. 1 (3)). As a result, messages are delivered only to the related nodes and the load of sender node is distributed to the other nodes. However, since ALM does not consider physical structure, inter-RDC communication increases unnecessarily (red arrows (a) in Fig. 1 (4)). The delivery considering physical structure can reduce inter-RDC communication (blue arrows (b) in Fig. 1 (4)).

In order to realize such a physical structure aware delivery, there are mainly two approaches, one is to optimize ALM using under layer information [3], [4], the other is to optimize under layer forwarding by directly controlling network equipment. In the following sections, those two approaches are introduced.

2.1 ALM Optimization Approaches

The issues in wide area deployment of IP multicast had already been recognized in 1990s and ALM development had been actively advanced [2]. In ALM, application layer end-systems exchange messages via overlay network constructed among them. The most important characteristic of ALM is that it does not require any new interaction among the operators of under layer infrastructures. It is easy to add new nodes to deploy ALM as a wide area service. Furthermore, ALM based on the structured overlay approach also has an autonomic resilience property in a fault situation because it does not require any centralized server. On the other hand, since ALM constructs overlay network independently of lower layer structure, unnecessary traffic generated by the overlay links go back and forth on the same physical links becomes a problem.

In order to solve the problem, overlay optimization methods are proposed [3], [4]. Issues in ISP related to the traffic increase caused by overlay network applications can be suppressed to a certain degree with those methods. However, they still have overheads and the possibility for further optimization.

2.2 Under Layer Optimization Approaches Including OpenFlow Multicast (OFM)

Recently, SDN technology including OpenFlow is the focus of network engineering field attention. It is possible to realize flexible interaction between application layer and lower layer by controlling lower layer function with modern style programming languages. Several methods have been proposed to establish flexible control over multicast network using OpenFlow functions [5], [6], which we call OpenFlow Multicast (OFM) in this paper. While they are the same as the IP multicast in the point that they realize multicast by lower layer functions, the difference is the centralized logic for targeting more flexible control. Those techniques can be used to reduce inter-broker traffic.

Furthermore, there is a new proposal of Pub/Sub architecture based on OpenFlow network, named PLEROMA [7]. It establishes Pub/Sub topic messaging in OpenFlow layer. As a result, it fully considers the physical layer structure to forward messages. However, it still has a scalability problem to support full functionality of the Pub/Sub system. OpenFlow Switch (OFS) looks up flow entries recorded at flow tables and forward packets based on the rules described in flow entries. Since flow entries must be processed in a short period, look up and processing logics are usually implemented using high cost devices like TCAM and thus the number of flow entries has an upper limit. While the number increased by technology progress year by year, it still has a limitation about 100 thousand entries per OFS even in the commercial products. Here, we consider a subscription of "send a message" in SNS, in other words, an event becoming a follower of a certain user in Twitter. In this case, for example, the number of events is considered to be in proportion to the number of accounts in Twitter or Facebook. It is reported as 271 million accounts in Twitter [8] and 1,280 million in Facebook [9]. Such a large number of events cannot be simply covered by the current OFS implementations. To establish Pub/Sub infrastructure based on SDN technology, further investigations must be required.

As one of the other approaches, there is a proposal of new generation Pub/Sub infrastructure based on the totally reconstructed lower layer functions [10]. It has a possibility to achieve even higher performance than OpenFlow based approaches. However, as discussed in the next section, it has a bigger problem than OpenFlow in the migration from the current network. In this paper, we focus on the OpenFlow as the more practical approach.

2.3 Hybrid Mode in OpenFlow

As a matter of course, the upper limit problem of flow entries is recognized in OpenFlow community. The solution to take advantage of OpenFlow in conjunction with the limitation is one of the important issues. Wackerly [11] proposes a method to save the number of flow entries by utilizing "Hybrid mode." Hybrid mode uses Pipeline Processing function standardized as OpenFlow switch specification [12]. It uses conventional lower layer functions, e.g., IP or Ethernet, implemented in the OFS. If we specify NORMAL action in a flow entry, the conventional lower layer functions forward packets instead of the flow base forwarding functions. It means that we can use flow entries only for more advanced and flexible requirements which are not supported by the conventional technologies. As a result, we can save the number of flow entries. The most of production OFS supports Hybrid mode, and it is expected as a way to take advantage of OpenFlow.

When we assume to use the Hybrid mode, the target network will be a mixed network of OpenFlow and conventional switches to support seamless migration from the conventional network. If we consider applying OFM to such a mixed environment, the same problem existing in the traditional IP multicast will occur again [2]. If OpenFlow networks are connected via conventional IP network, tunneling function to connect partial OFMs on separated OpenFlow networks is required in OFM constructions. When OFSs become widespread and all of the networks are OpenFlow ready, the tunneling function becomes unnecessary. However, when we construct wide area OFM network, it spreads over multiple OpenFlow administrative domains. It is still difficult to extend control plane of OpenFlow network over several administrative domains with different operational policies.

Here, if we consider the possibility of ALM again. IP reachability among brokers becomes a minimum requirement for supporting ALM and Hybrid mode can satisfy the requirement. From the above consideration, combination of ALM and OFM is a way to take advantage of OFM with Hybrid mode.

While Pub/Sub with ALM and OFM have a possibility to scale Pub/Sub infrastructure, either of the multicast methods alone still has issues as described in Sections 2.1 and 2.2. How to federate those methods to complement each other is the key to establish scalable Pub/Sub system.

3. Scalable Pub/Sub System Using OpenFlow Control

In order to federate multicast methods as described in Section 2, we proposed a basic idea of a hybrid architecture which exploits both merits of ALM and OFM to handle massive amount of traffic [13], [14]. The purpose of this study is to describe a design of Scalable Pub/Sub system using OpenFlow Control named as SDN Aware Pub/Sub (SAPS) which is based on the concept proposed in Refs. [13], [14] and to evaluate it by simulations. The overview of SAPS is shown in **Fig. 2**. Host A, B, C are hosting broker instances and those hosts are connected to OpenFlow backbone network. P2P overlay network is also constructed among broker instances and it provides ALM function by using topics and agents as described later. In order to reduce inter-broker traffic, not only ALM but also OFM is exploited in SAPS if OFM is ready for the backbone network. In this simple example network, all brokers are ready to use OFM.

The "topic-based Pub/Sub" is adopted in SAPS because it is adopted by major Pub/Sub products. Since the brokers must communicate flexibly with each other in SAPS, we assume P2P agent platform (P2PAP) like PIAX [15] as the basis of the brokers. Here, broker is the same as peer in P2PAP. P2PAP provides flexible communication capability using Agent API as described in **Fig. 3**. Each agent can register its attributes into overlay network. In Fig. 3, agent 1 registered "attr1" and "attr2." Those attributes are used in Agent API to choose callee agents. For example, *discoveryCall* (*attribute, method, args*) API is used to call a method of an agent and if an attribute matching multiple agents is specified, e.g., "attr3," it can be used as ALM. Agent API utilizes structured overlays like DHT [16] and SkipGraph [15] for

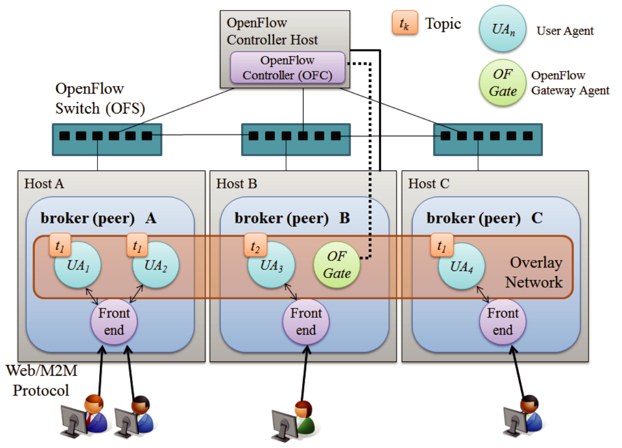


Fig. 2 Overview of Scalable Pub/Sub system using OpenFlow Control (SAPS: SDN Aware Pub/Sub).

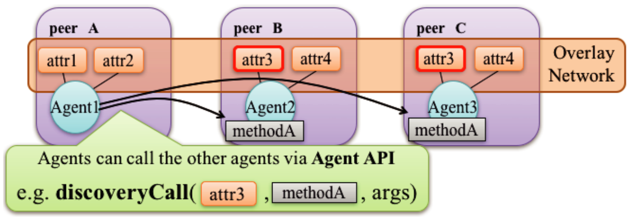

peer A
peer B
peer C
attr1
attr2
attr3
attr4
Overlay Network
Agent1
Agent2
Agent3
methodA
Agents can call the other agents via Agent API
e.g. discoveryCall(attr3 , methodA , args)

Fig. 3 PIAX Agent API example.

assuring message arrival. An attribute of an agent is used as a topic t_k as describd in Fig. 2 and also the name of the agent to call its methods.

Two agents play important roles in SAPS. One is *User Agents* (*UAs*) and the other is *OpenFlow Gateway Agent* (*OFGate*) as described in Fig. 2. In SAPS, The brokers mediate publishers and subscribers as usual Pub/Sub system. When a publisher/subscriber connects to the brokers via FrontEnd using Web or M2M protocols, a *UA* is generated corresponding to the publisher/subscriber. From now on, since we mainly discusses about inter-broker communications, we call the *UA* on a broker as "publisher agent" or "subscriber agent," which is mapped to the real publisher or subscriber.

In order to control OpenFlow network from the brokers, they must communicate with OpenFlow Controller (OFC). *OFGate* is an agent prepared for realizing interaction between ALM and OFM. *OFGate* is generated at the broker which can communicate with OFC using Northbound API as described in Section 3.1.1. In Fig. 2, since the control plane is designed as out-of-band style to satisfy the security requirements or protect control traffic, *OFGate* must be located at Host B which has a connection to OFC. If in-band style is adopted and any node can access to OFC, *OFGate* can be freely located. Since any agent can communicate each other using Agent API, we assume that publisher agents and subscriber agents can communicate with *OFGate* in the following descriptions.

Publisher agents can also receive responses from subscriber agents, if needed, by specifying in the message. The response is assumed to be aggregatable by tracing the multicast path in the reverse direction. Publisher agents collect monitoring information by using that function when they publish a message.

In the rest of this section, we will discuss how to exploit OFM in addition to ALM. In SAPS, a message published as a new topic is delivered via ALM, and then by referring the monitoring information, it is decided to exploit OFM or not. The decision is made by considering the number of delivered messages, the size of messages, delivery frequency, a priority value set to the messages and so on. Once the exploitation of OFM is decided, an OFM mapped from the ALM of the specified topic is constructed in an OpenFlow network.

In order to provide switching function between ALM and OFM, we need to consider the API of SAPS. In Section 3.1, we describe the design of SAPS APIs, the one exists between *OFGate* and OFC and the other exists between SAPS and the end system. Furthermore, when we exploit the both ALM and OFM, we need to synchronize the membership management of the both multicast network. The details are discussed in Section 3.2.

3.1 SAPS API

To realize the interaction between *OFGate* and OFC, we need to design an API of OFC, called Northbound API. And we also need Pub/Sub API as the same as the conventional Pub/Sub, e.g., publish/subscribe, which must be transparent to the switching between ALM and OFM. **Figure 4** shows relationship between system components and APIs. In Section 3.1.1, a design of Northbound API is described and then in Section 3.1.2, the Pub/Sub

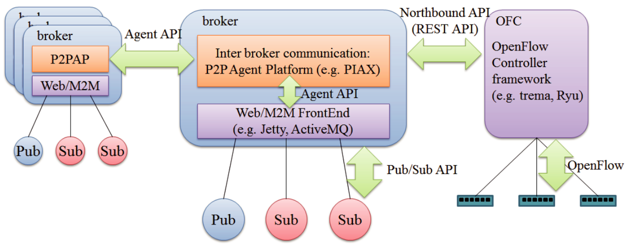

broker
P2PAP
Web/M2M
Agent API
broker
Inter broker communication: P2P Agent Platform (e.g. PIAX)
Agent API
Web/M2M FrontEnd (e.g. Jetty, ActiveMQ)
Northbound API (REST API)
OFC
OpenFlow Controller framework (e.g. trema, Ryu)
Pub/Sub API
OpenFlow
Pub
Sub
Sub
Pub
Sub
Sub

Fig. 4 Relationship between system components and APIs.

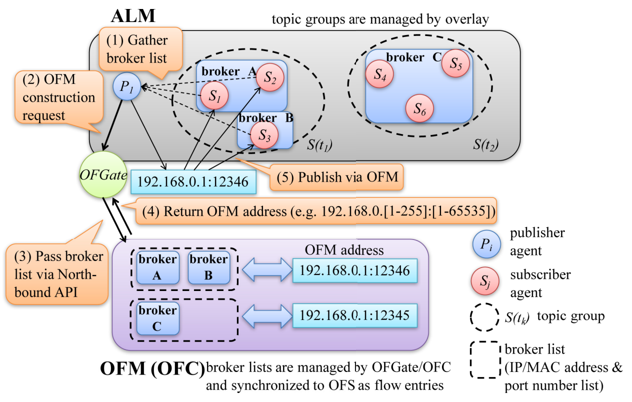

ALM
topic groups are managed by overlay
(1) Gather broker list
(2) OFM construction request
P_1
broker A
S_1
S_2
broker B
S_3
$S(t_1)$
broker C
S_4
S_5
S_6
$S(t_2)$
OFGate
192.168.0.1:12346
(5) Publish via OFM
(4) Return OFM address (e.g. 192.168.0.[1-255]:[1-65535])
(3) Pass broker list via Northbound API
broker A
broker B
broker C
OFM address
192.168.0.1:12346
192.168.0.1:12345
OFM (OFC)
broker lists are managed by OFGate/OFC and synchronized to OFS as flow entries
P_i publisher agent
S_j subscriber agent
$S(t_k)$ topic group
broker list (IP/MAC address & port number list)

Fig. 5 Overview of ALM/OFM mapping.

API for SAPS is described.

3.1.1 Northbound API

Figure 5 shows the mapping between ALM and OFM during the OFM construction. The top of the Fig. 5 shows the status of ALM. A subscriber agent is registered and managed as a member of a topic group by registering a topic key to the overlay, which we call "subscribe" in ALM. In Fig. 5, subscriber agent S_1, S_2 and S_3 are the member of the topic group $S(t_1)$ which is a group of subscriber agents interested in the topic t_1. During the subscription, publisher agents cannot detect the member changes because the overlay network is updated locally around the new subscriber agent. As described later, when OFC creates a multicast tree related to the topic t_k, it requires the list of the brokers hosting the subscriber agents in the topic group $S(t_k)$. How to collect *broker list* at *OFGate* is described in Section 3.2. An element of *broker list* is described as a tuple <IP address, MAC address, port number>. In order to provide flexibility in address assignment by using OFM address, SAPS rewrites the destination MAC address to the target node address described in the broker list.

A publisher agent makes the decision of OFM construction based on the monitoring information and sends a request to *OFGate*. After receiving the request, *OFGate* asks OFC to construct OFM via REST API. The OFC assigns *OFM address* to the *broker list* and returns it to *OFGate*. *OFM address* is composed of IP address and port number like "<IP address>:<port number>," and its range must be preregistered to the system. This approach has a merit of flexibility compared to the use of the standardized multicast address. After finishing the construction of OFM, the publisher agent publishes a message to the subscriber agents by sending packets to *OFM address*.

OFC construct multicast tree for packet delivery from a publisher agent to subscriber agents by using *broker list*. There are mainly two approaches to construct OFM tree, one is to construct source specific tree and the other is to construct shared tree [17]. While the source specific tree approach must be adopted if the performance is the principal factor, it requires to construct multicast tree for each publisher agent. When the number of publisher becomes large, the flow entry limitation must be a problem as described in Section 2.2. In this paper, since we assume IoT and SNS applications where an arbitrary end entity becomes a publisher, we need to consider the shared tree approach. In order to support publish from subscriber agents, bidirectional tree is constructed among subscriber agents.

3.1.2 Pub/Sub API

SAPS provides basic interface, e.g., publish/subscribe, like general Pub/Sub system. While the API is the same as usual, the behavior inside the broker changes for each multicast method. For example, when a publisher agent publishes a message to ALM, it is confirmed by transport layer function, like TCP ACK. However, when the publisher agent uses OFM, it basically uses UDP and the publisher agent cannot confirm the packets are properly sent via OFM or not. In order to enable publisher agents to confirm the existence of OFM, the packets sent by publisher agents are sent back with rewriting source and destination by adding flow entries. Flow entries for that purpose are also set for the interfaces where subscriber agents are connected because in many cases, a subscriber also becomes a publisher as assumed in Refs. [18], [19]. The packet sent back to a publisher agent is named as “loopback packet.” Furthermore, the message arrival can also be ensured with reliable multicast technique [20], [21]. Since in many cases, the number of subscriber agents is larger than that of publisher agents, it is desirable that subscriber agents receive messages regardless of the multicast method selected by the publisher agent, ALM or OFM. Therefore, the broker where subscriber agents exist listens on the both ALM port and OFM port, and then the broker delivers the received message to the subscriber agents by topic name written in the message. Since the broker manages mapping between subscriber agents and topics, the only one listen address and port is enough for ALM and OFM respectively.

In addition to constructing multicast tree, the OFC must generate the flow entries which rewrite the destination address of the packets from *OFM address* to the listen address and port of the broker with subscriber agents.

In the rest of this paper, “topic” string is often used to manage ALM and OFM. However, the descriptions added by SAPS are not necessary to be shown to the end users. The end users just express the interested topics as a string. The additional descriptions are automatically added or deleted in SAPS.

3.2 Membership Management and Delivery Method

As described in Section 3.1.1, in order to construct OFM including ALM subscriber agents, it is necessary to collect *broker list*, and it must be notified to the OFC. However, if a subscriber agent joins or leaves after the collection and before the completion of OFM preparation, an inconsistency is generated between the *broker list* on OFC and topic group on ALM. As a result, messages cannot be delivered to some of the subscriber agents. Such a message loss must be avoided. Furthermore, we have choices in multicast delivery methods, the one is utilizing either ALM or OFM (EITHER) and the other is utilizing the both ALM and OFM (BOTH). The details of the two methods are described in the following sections.

3.2.1 Utilize Either ALM or OFM (EITHER)

Firstly, we propose a delivery method which uses either ALM or OFM (EITHER). The following approach is taken in EITHER to establish strict membership management.

- a subscriber agent notifies subscription to *OFGate* after joining ALM
- a publisher agent asks *OFGate* about the OFM construction status before publish

When a publisher agent decides to exploit OFM based on monitoring information, it asks *OFGate* to construct OFM. In EITHER, since subscriptions are notified to *OFGate*, *broker list* already exists at *OFGate* and the process (1) in Fig. 5 is not required.

If a publisher agent asks *OFGate* about the OFM status during OFM construction, *OFGate* returns the status of “under construction.” In this case, the publisher agent uses ALM. When a construction finished, *OFGate* returns the status of “constructed” and the publisher agent uses OFM. When a subscriber agent joins or leaves, the update request is sent to *OFGate*. During the update, *OFGate* returns “updating” and the publisher agent uses ALM. If the update request arrives during OFM construction, the exploitation of OFM is delayed until the update completion. When the all of the subscriber agents have been left from a topic, the related OFM is deleted. During the deletion, the status is managed in the same way. Publisher agents can start exploiting OFM just after the completion of construction because it confirms the status before publishing.

The merit of this method is as follows:

- It is simple to implement.

The demerits are as follows:

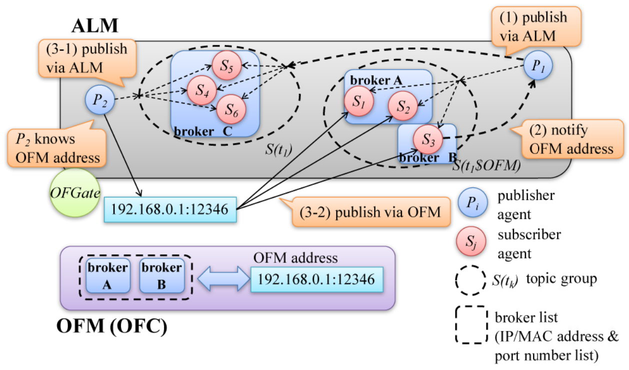
- *OFGate* must deal with heavy query load
- Since the broker list is synchronized to topic groups, once the OFM is constructed, all of the updates must be reflected to the OFM.

An approach to exploit EITHER method is to cache OFM address locally at the publisher agent. It can reduce the load of *OFGate*. However, if publisher agent uses cached OFM address during OFM updates, it may cause message loss. Join and leave frequency should be considered to set cache expiration.

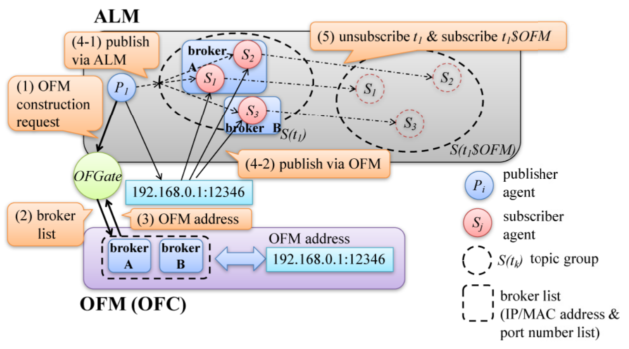
Furthermore, if subscribers join and leave in high frequency during the switching procedure between ALM and OFM, it will cause a certain amount of overhead. In EITHER case, agents can start to use OFM just after completion of OFM construction. However, if the joining and leaving requests are queued into *OFGate* during the OFM construction, the activation time of OFM is delayed until those requests have been properly processed. Though the processing time of each joining and leaving request is smaller than OFM construction, if the number of requests becomes extremely large, it is possible to reduce the availability of OFM. About the leaving of subscribers, the activation time of OFM can be hastened if it does not wait for the finishing time of leaving procedure. It means that the wasteful messages are delivered to the brokers without subscriber agents. Which method performs better depends on the frequency of joining/leaving and the focusing parameters, e.g., number of messages and message delay.

3.2.2 Utilize Both ALM and OFM (BOTH)

Secondly, we propose another delivery method which uses the both ALM and OFM (BOTH) in a delivery. In BOTH, *broker list* is managed in overlay network. There are two types of *broker list* related to topic groups $S(t_k)$ and $S(t_k\$OFM)$. $S(t_k)$ means subscriber agents who are interested in topic t_k and receive the messages via ALM. On the other hand, $S(t_k\$OFM)$ means subscriber agents who are also interested in topic t_k and receive the messages via OFM. By managing two *broker lists* separately, it enables us to use ALM and OFM for a delivery simultaneously. Furthermore, since ALM and OFM membership is managed in overlay network, there is no notification to *OFGate* on publish or subscribe. It is different from EITHER. Therefore, the publisher agent must collect a *broker list* as a part of monitoring information, and then if it decides to construct OFM, it passes the *broker list* to *OFGate*. An example flow of BOTH is shown in **Fig. 6**. Figure 6 (a) shows an example of publish flow. If a publisher agent does not know *OFM address*, as P_1 described in Fig. 6 (a), the publisher agent publishes to the both of $S(t_1)$ and $S(t_1\$OFM)$ via ALM. At this time, subscriber agents in $S(t_1\$OFM)$ return *OFM address* to the publisher agent. As a result, the publisher agent can get *OFM address* without querying *OFGate*, and after that it can publish via OFM. While the OFM address returned by all the subscriber agents in $S(t_1\$OFM)$, the messages are aggregated into one message as described in Section 3. If a publisher agent already knows *OFM address*, as P_2, it publishes to $S(t_1)$ via ALM (3-1) and $S(t_1\$OFM)$ via OFM (3-2). If the publisher agent does not receive loopback packet from the OFM, it recognizes that the OFM was deleted and it delete *OFM address* from its own memory. Here, if $S(t_1)$ is empty, publish via ALM does not send any packet to the network.



(a) Publish example with ALM and OFM subscriber agents



(b) An example of subscriber agents' topic migration

Fig. 6 Example flows in BOTH method.

Figure 6 (b) shows an example that the publisher agent P_1 decides to construct OFM. After the completion of OFM construction, the publisher agent publishes via ALM (4-1) and via OFM (4-2). At this time, since subscriber agents receive messages from the both of ALM and OFM, they must resolve the duplication by using message ID assigned at the publisher agent. Subscriber agents belonging to $S(t_1)$ receive messages via OFM and noticed that they are ready to receive messages via OFM. Therefore, the subscriber agents subscribe $t_1\$OFM$ and then unsubscribe t_1. Here, since the simultaneous subscribe/unsubscribe of the all subscriber agents causes churn, the subscriber agents wait random period, and then execute subscribe/unsubscribe operations. In the case that a subscriber agent unsubscribes $t_1\$OFM$, the request must be sent to *OFGate* to update OFM.

The merits of this method are as follows:

- The number of query to OFGate is reduced
- Partial migration to OFM is supported

The demerits of this method are as follows:

- During the migration to OFM, message duplication occurs
- After OFM construction, publisher agents without *OFM address* still send messages via ALM

The second demerit means that after OFM construction, the performance of BOTH becomes worse than EITHER until all publisher agents obtain OFM address.

About the performance of frequent joining and leaving of subscriber in BOTH, since newly subscribed agents will join ALM, the performance depends not on OFM but on ALM performance.

3.3 Monitoring Function

In SAPS, system monitoring plays an important role to decide which multicast method should be used. In this section, we will discuss what kind of information is monitored and how to collect the information in SAPS.

Number of published messages, publish frequency and message size per topic are candidates of simple monitoring information. Those statistics are used as parameters in the evaluation in Section 5. They can be recorded at subscriber agents on message arrival and can also be used without message exchanges among agents. As described in Section 3, since ALM is used for the membership management in SAPS, a publisher agent can gather statistics as aggregated reply messages from subscriber agents via distribution tree in the opposite direction. It can be used as a simple collection method for monitoring information. Furthermore, it is also possible to collect network device statistics by OpenFlow function via *OFGate*. However, there exists a trade-off between frequency of collection and accuracy of monitoring information. The evaluation of monitoring cost should be required as a part of future work.

3.4 Security and Fault Tolerance

In this section, security and fault tolerance related topics are discussed.

3.4.1 Security

As described in Section 3, SAPS can interact with end user/system via Web or M2M protocols. To establish access control with end entities, authentication and authorization technologies in the Web or M2M fields can be applicable. Here, back-end side brokers are assumed to be trustable with each other in SAPS. Inter-broker authentication and message encryption are realized by the keys which are distributed with the broker software in deployment phase. Based on the assumption, the type of cryptography and key management method among brokers can be freely chosen, e.g., symmetric-key or public-key cryptography, ID-based cryptography [22], pre-shared key, public key infrastructure (PKI) and so on, based on the requirements in the deployment.

In order to establish authentication and encryption for UDP multicast used in OFM, the group key must be shared among the subscriber agents of the same topic. Efficient group key sharing method with a large number of group members are already proposed in Ref. [23] and applied to the encryption of IP multicast [24], [25]. In SAPS, the same approach can be taken to implement authentication and encryption for multicast packets. The method proposed in Ref. [23] still has several limitations as described in Section 6.2 of Ref. [24]. For example, it is difficult to defend DoS attack to the multicast address. In order to establish more secure environment, network space separation of Pub/Sub infrastructure as proposed in Ref. [26] must be required.

3.4.2 Fault Tolerance

The fault tolerance of the proposed system is established by the feature of Hybrid mode. Since the conventional IP network is constructed in a distributed manner, even if it is divided into multiple portions by a fault, it can still tolerate the situation. Here, if we implement multicast only by the current OpenFlow functions, we cannot fully exploit the fault tolerance. For example, as Tootoonchian [27] proposed, deploying multiple OFC is an approach to support fault tolerance in OpenFlow. However, it is less tolerable than IP network unless all the OFS have their own controller. While there is a work on the problem [28], it still requires further research. Therefore, we currently focus on the fault tolerance exploiting OpenFlow-hybrid functions.

3.5 Comparing OFM and IP Multicast

SAPS adopts OFM as a lower layer multicast. However, IP multicast also has a possibility to be utilized. In this section, we compare OFM with IP multicast.

The basic merit of OFM is described in Section 2.2 that is centralized and flexible control. Even if the target network is a mixed network of OpenFlow and conventional switches, that merit is also effective in each OpenFlow switch segment. In the case where OpenFlow switches implement IP multicast as conventional switch function and only simple IP multicast function is required by the applications, IP multicast is easier to be deployed to overall network.

However, SAPS has the following advantages compared to the system with IP multicast. SAPS can provide (1) closer connection between network layer and application layer, and, (2) a relaxation of IP multicast address limitation. Those features are not provided by IP multicast and it is worth to be exploited in spite of its deployment cost.

End user activity monitoring can be considered as an example exploitation of advantage (1). A question "What kind of topics are popular recently?" can only be answered by application program because it is difficult to extract application layer knowledge by network layer monitoring. On the other hand, the network layer information can only be monitored by network devices. If we control both application and network interactively based on such kind of information, the API like Northbound API of OpenFlow is required. Since the conventional IP multicast only provides joining and leaving interface to the applications, it is necessary to extend switch function for controlling network based on the application layer information.

About the advantage (2), SAPS generates OFM address by combining IP address and port number. As a result, it enables to use address range more flexibly than IP multicast where predefined multicast address is used.

From the above consideration, OFM has merits to be adopted in SAPS.

4. Evaluations and Considerations

In order to confirm the effectiveness of hybrid approach in SAPS, it is necessary to compare the performance of ALM and OFM. In this section, we describe scalability evaluations. One is a simulation of traffic and message transmission delay of ALM compared with OFM. The other is analyzing the number of flow entries required for OFM as discussed in Section 2.2. We will show the evaluation results in Section 4.1 and discuss the issues remaining in the current design including exploitation approaches of SAPS architecture in Section 4.2.

4.1 Scalability Evaluation

In this section, we show scalability evaluation results. In the evaluations, we define the "cluster" as a set of the broker hosts mutually connected by switches with enough bandwidth, which means that each switch can transfer any size of message with specified number without packet losses, and then connect clusters each other by switches as shown in **Fig. 7**. In Fig. 7, a host

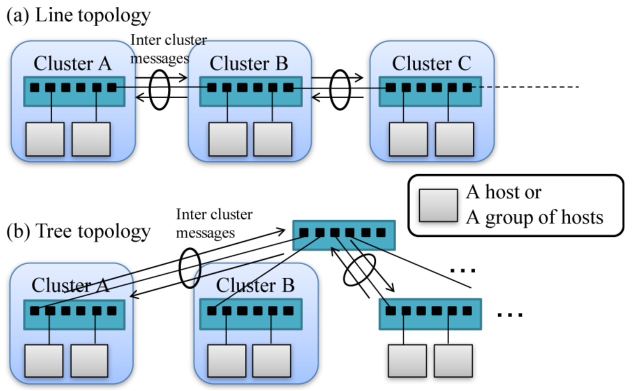


Fig. 7 Simulation topology.

or a group of hosts inter-connected by layer 2 or 3 switches are drawn as a box. We evaluated the reduction performance of inter-cluster traffic and message transmission delay. Since a message distributed by OFM is copied at the switches, a message sent to and from a switch is counted separately. As a result, a message sent from a cluster to the other cluster by unicast is counted as two messages. The parameters are the number of clusters (4, 8, . . . , 128), the topology (line, tree) and the number of subscribers. The practical data center network topology is described in Ref. [29] and it has intermediate characteristic of line and tree topology. So, as a first step, we examine those two patterns. As an overlay network for ALM, SkipGraph, which has relay-free characteristics when the overlay is constructed using topics as keys, is used [30]. During the construction of SkipGraph overlay, since membership vectors are generated as random values, simulations are executed with 100 different overlays for each parameter set. As described in Section 2, ALM forwards messages without considering the lower layer topology, in some cases, messages go back and forth on the same physical link. However, if it is possible to obtain the cluster ID where the broker is belonging to and topics can be sorted by cluster ID, e.g., t_1*@clusterA*, t_1*@clusterB*, unnecessary inter-cluster communications can be reduced because SkipGraph creates logical links based on the sorted order of registered keys. In this case, cluster ID helps to create logical links between the brokers located at the same or neighboring cluster. As a result, unnecessary jumps between distant clusters are reduced. In the rest of paper, we call the method using cluster ID as ALM with Locality Awareness (ALM-LA) and add as a target of comparison. Since the details of Locality Awareness are discussed in Ref. [31], we adopt simple method in this paper. In the simulation, subscribers are placed to the hosts with a uniform distribution. About the OFM related parameters, the OFM constructing/updating delay is omitted in this evaluation to investigate ideal traffic reduction in OFM. The results are shown in the following sections.

4.1.1 A Simulation of Traffic and Transmission Delay

The simulation results are shown in **Figs. 8**, **9**, **10**. Figure 8 shows results of line topology with 16 clusters. The left side shows the inter-cluster traffic per publish and the right side shows the maximum delay (cluster hops) per publish. In the both results, OFM shows better performance than ALM. When the cluster ID is numbered randomly, ALM-LA performance obviously degrades. Especially in the maximum delay, it is almost 4 times as large as the sorted case. If a pair of clusters is randomly chosen from line topology, the hops between the two clusters become far larger than the sorted case. It causes performance degradation. On the other hand, the physically sorted result shows better performance. From those results, it is said that if an identifier which is assigned regardless of physical order, e.g., IP address, is used as a sort key of ALM-LA in line topology, OFM efficiently improve the performance. Figure 9 shows results of height 2 tree topology with 16 clusters. Since inter-cluster distance is always 2 hops in the tree topology, the all methods show slightly better performance than in the line topology. In this evaluation, clusters are randomly located because the location does not affect the performance. In the tree topology, even if it has only

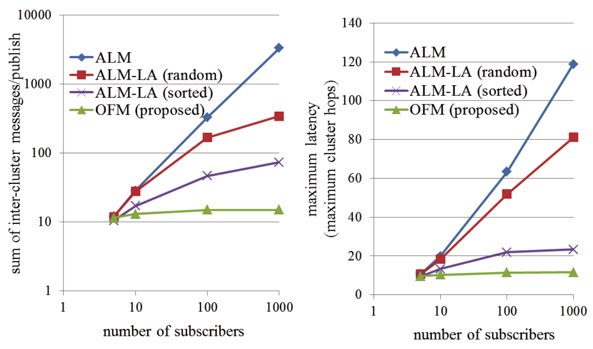


Fig. 8 Line topology (16 clusters) results.

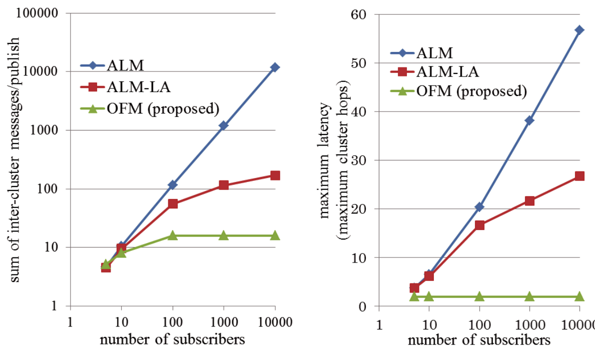


Fig. 9 Tree topology (16 clusters) results.

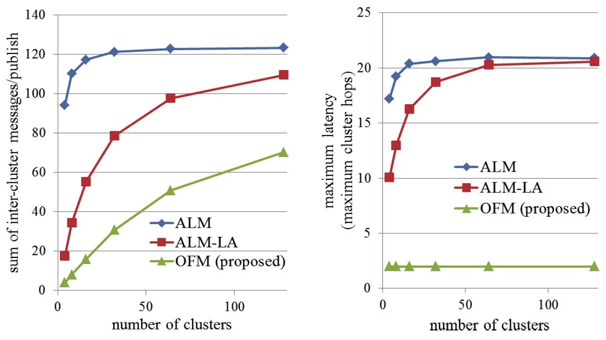


Fig. 10 Tree topology (4–128 clusters) results.

100 subscribers, OFM can reduce inter-cluster traffic 71.6% with 16 clusters compared to ALM-LA. It can also reduce maximum inter-cluster hops 87.5%.

Figure 10 shows results of height 2 tree topology varying in number of clusters from 4 to 128 with 100 subscribers. When the number of clusters is small, since all of clusters have at least one subscriber, OFM sends one message per cluster and the number of messages is in proportion to the number of clusters. On the other hand, the maximum inter-cluster hop is always 2 as the same as the above result. As the number of clusters increases, the number of subscribers per one cluster decreases. That causes the number of messages of OFM to be smaller than the number of clusters. For ALM and ALM-LA, that causes most of the logical hops between SkipGraph nodes to be inter-cluster hops. As a result, the both results of ALM and ALM-LA approach the performance of SkipGraph with the larger number of clusters. It means that OFM still shows better results in large scale cluster environment.

From the results shown in this section, OFM has the efficiency to reduce traffic and delay with only 100 subscribers. For example, in IoT applications, we can set topics for each room or building where target sensors are located. Limiting the number of subscribers and related area enable us to extract stable and closely connected sub groups. The problem of EITHER method can be reduced in such a situation.

4.1.2 Analyzing Resource Consumption in OFM

From the simulation results, OFM is superior to ALM. On the other hand, as described in Section 2.2, the number of flow entries becomes an obstacle in applying OFM for all topics. **Figure 11** shows an example flow entries for an OFM tree described in Section 3.1.1. Red arrows indicate packet forwarding rule among OFS. When an OFM packet comes into the OFS, it should be copied and forwarded to the related ports except for the input port. Blue arrows show address conversions required before packet output. Since a set of ports can be represented as Group from OpenFlow 1.1 [12], the red arrows in one OFS can be implemented as 1 flow entry. And the blue arrows become group action buckets of the flow entry. Since the actions must be different for each switch port, it must be considered to require additional OFS resources. In order to consider the resource consumption by OFM, the number of flow entries and group action buckets in the previous simulation situation are analyzed in **Fig. 12**. The left side shows the ratio of number of flow entries to total number of switches and the right side shows the ratio of number of group action buckets per port to total number of switches. Those ratios indicate that when an OFM tree related to a topic constructed, how many switches must consume resources. In the analysis, subscribers are placed by uniform distribution and the maximum number of subscribers per host is assumed to be 1,000. In the left graph, with 100 subscribers, all of the switches must have at least one flow entry for an OFM tree. We also examine Zipf distribution [32] in tree topology cases to simulate the scenarios in which the popularity of subscribers is concentrated within a small number of clusters. The result still shows that 1,000 subscribers are enough to involve all of the switches. In the right graph, as the same as flow entries, group action buckets distribute all the edge switches with 100 subscribers. While the rate becomes larger than 1.0 with more than 10,000, it depends on the maximum number of subscribers per hosts. From those results, when an OFM tree constructed for a topic, it is said that almost all the switches are involved in the construction. In other words, even the edge switches must have enough resources to host all topics as OFM trees. The resource limitation becomes more critical when we consider more optimized OFM trees, for example, optimal trees for each publisher. The number of publishers becomes as large as the number of subscribers when we assume many-to-many communication like SNS. Reference [33] reports 13% of twitter users actively publish their tweets. In this case, 10–20% of subscribers are assumed to be publishers. Finally, the most critical topics must be chosen for optimization that is to say the hybrid approach should be applied.

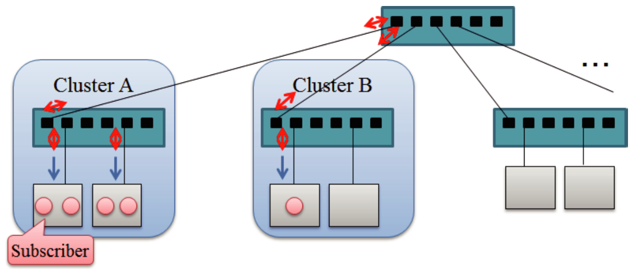


Fig. 11 An example flow entries for OFM.

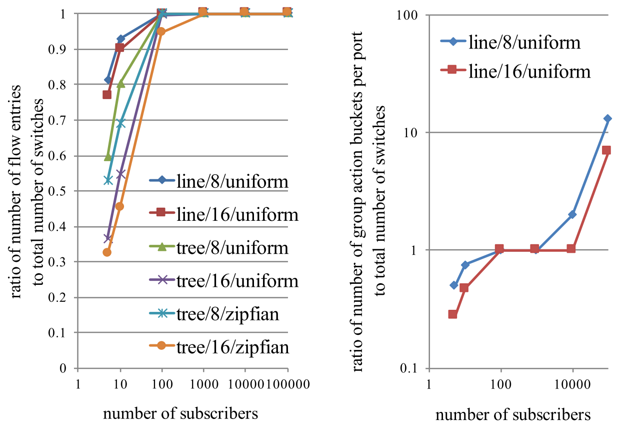


Fig. 12 Ratio of flow entries and action entries.

4.2 Comparing Number of Queries in EITHER and BOTH

The proposed membership management and delivery methods EITHER and BOTH have merits and demerits.

The number of queries sent to *OFGate* per unit time for EITHER and BOTH are expressed as Q_{EITHER} and Q_{BOTH} respectively in the following equations. The parameters are explained in **Table 1**.

$$Q_{EITHER} = P_n \cdot P_r + S_n \cdot S_l + S_n \cdot S_j \quad (1)$$

$$Q_{BOTH} = S_n \cdot S_l \quad (2)$$

In EITHER, since agents are necessary to query *OFGate* on their publish, subscribe and unsubscribe, Q_{EITHER} becomes as Eq. (1). In BOTH, since agents only queries on their unsubscribe, Q_{BOTH} becomes as Eq. (2).

In **Fig. 13**, the number of queries submitted to *OFGate* is plotted for EITHER and BOTH by changing the rate of joining/leaving subscribers under the environment where publisher

Table 1 Parameters used in Eqs. (1) and (2).

P_n	number of publisher
P_r	number of publish per unit time
S_n	number of subscribers
S_l	rate of leaving subscribers per unit time against S_n
S_j	rate of joining subscribers per unit time against S_n

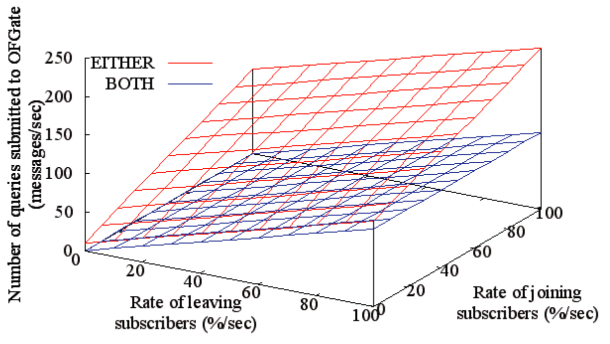


Fig. 13 Number of queries submitted to OFGate.

publish one message per second. z-axis shows number of queries submitted to *OFGate* per unit time and x-axis and y-axis shows rate of joining and leaving subscribers per unit time. The number of subscribers and publishers are set to 100 and 10 respectively. As shown in Fig. 13, EITHER always shows larger number of queries than BOTH because Q_{EITHER} is larger than Q_{BOTH} by the number of publishes from publishers and joining subscribers per unit time. From the result, if we simply consider the number of queries to *OFGate*, BOTH shows better results. However, in the case that the influence of the delay for OFM address notification to publisher agents in BOTH is larger than the overhead of queries to *OFGate* in EITHER, The latter shows better results. Since that point depends on the the frequency of OFM construction, we need to define an indicator to switch ALM and OFM which requires an evaluation on the prototype system. After that, detailed comparison should be executed.

From the results, the amount of traffic in the control plane and requests received at OFC can also be estimated as against the frequency of publish, subscribe and unsubscribe. The number of control message sent to the control plane depends on type of query, number of switches and network topology. When an OFM construction request is received as a query, the control messages must be sent to the whole switches involved in connecting brokers each other. If the query is a join/leave request, the messages are sent only to the switches related to the broker where the target agent resides or no message is sent when the other subscriber agents still remain in the same broker. In order to show the frequency of OFM construction requests, we need to define an indicator to switch ALM and OFM which requires an evaluation on the prototype system as described above. The concrete load of OFC also requires a measurement on the system. We will tackle them as a part of future work.

5. Conclusions and Future Work

In this paper, we have picked up several new demands for Pub/Sub infrastructure and in order to meet the demands, we designed the basic functions of Scalable Pub/Sub system using OpenFlow Control (SAPS). In addition to ALM, SAPS exploits OFM to reduce traffic and message transmission delay for inter-broker communication. A simulation was done for evaluating the hybrid architecture in traffic and transmission delay reduction. The result shows that in the tree topology, even if it has only 100 subscribers, OFM can reduce inter-cluster traffic 71.6% with 16 clusters compared to ALM-LA. It can also reduce maximum inter-cluster hops 87.5%. On the other hand, with only 100 subscribers, almost all of switches are involved in the OFM tree construction and consume flow table space for the topic. It indicates that our hybrid approach is effective in Pub/Sub optimization considering the resource limitation of OpenFlow switches.

Finally, we mention future work on SAPS. We have already implemented a prototype system. However, its evaluation has not finished yet. As a part of future work, we will evaluate total system performance including the investigation of ALM maintenance cost, ALM and OFM switching cost, churn resiliency and so on.

The destination address rewriting of SAPS discussed in Section 3.1.2 provides flexibility for address usage. However, currently, the address rewriting in OFS is known to have performance issue. It should also be evaluated in the prototype system.

SAPS maps application layer topics into network layer identifier, OFM address. It enables us to give priority to specific topics, e.g., emergency information delivery in SNS [34], not only at the end node but also inside the networks. The Pub/Sub protocol like MQTT already has QoS properties for server-client communication. We will also consider the QoS properties in SAPS.

The method proposed in this paper currently support only single administrative domain. Hybrid use of ALM and OFM which spreads over multiple domains including domain splitting as described in Section 2.3 is a part of future work. However, we already provide materials to consider the extensions for multiple domain awareness. For example, as discussed in Section 4.1, cluster ID can be used to map OpenFlow administrative domain and overlay network, e.g., topic1@domain1, topic1@domain2. Once the domain is mapped to the overlay network, subscriber agents can join the topic with local domain. At the same time, *OFGate* can also be prepared for each domain. It enables us to switch OFM utilizing local domain function. Furthermore, it is possible to elect a representative subscriber agent for each domain and the representative subscriber agent forwards messages to OFM which were received from the other domains. The detail design and implementation of multi-domain function is future work.

As already discussed in Section 3.4.1, how to deploy SAPS in the practical environment is one of important issues. When it is used only with ALM function, SAPS can be used as a Pub/Sub middleware by providing its codes as an open source product. The application developers use basically ALM and if they need an assurance of network performance or quality, activate *OFGate* function. In this case, the application developer or the application service provider should agree with a quality assurance contract with Data Centers or ISPs. For example, DCs or ISPs provide some metadata file, and install the file into SAPS, *OFGate* automatically startup and using license key provided with the metadata file to connect Northbound API of DCs or ISPs infrastructure. Another approach to utilize SAPS is by applying its technology in Pub/Sub service provider. It is like a CDN service provider, such as Akamai. Such a deployment approach must be considered with the future development plan.

Acknowledgments This study is partially supported by MIC/SCOPE #150201013, FESTIVAL project under FP7-ICT-2013-EU-Japan collaborative project and National Institute of Informatics's joint research project. We would also like to express our sincere thanks to the late Mr. Mikio Yoshida for giving us advanced and discerning advices to design the proposed architecture.

References

[1] Hernández-Muñoz, J.M. et al.: Smart cities at the forefront of the future internet, Springer Berlin Heidelberg (2011).
[2] Hosseini, M., Ahmed, D.T., Shirmohammadi, S. and Georganas, N.D.: A survey of application-layer multicast protocols, *IEEE Communications Surveys & Tutorials*, Vol.9, No.3, pp.58–74 (2007).
[3] Xie, H., Yang, Y.R., Krishnamurthy, A., Liu, Y. and Silberschatz, A.:

P4P: Provider portal for applications, *Proc. ACM SIGCOMM 2008*, pp.351–362 (Aug. 2008).
[4] Seedorf, J., Kiesel, S. and Stiemerling, M.: Traffic localization for P2P-applications: The ALTO approach, *IEEE P2P 2009*, pp.171–177 (Sep. 2009).
[5] Kotani, D., Suzuki, K. and Shimonishi, H.: A design and implementation of OpenFlow controller handling IP multicast with fast tree switching, *Proc. IEEE/IPSJ 12th International Symposium on Applications and the Internet* (*SAINT*) (2012).
[6] Marcondes, C.A.C. et al.: CastFlow: Clean-slate multicast approach using in-advance path processing in programmable networks, *Proc. IEEE Symposium on Computers and Communications* (*ISCC*) (2012).
[7] Tariq, M.A., Koldehofe, B., Bhowmik, S. and Rothermel, K.: PLEROMA: A SDN-based high performance publish/subscribe middleware, *Proc. 15th International Middleware Conference* (*Middleware '14*), pp.217–228 (Dec. 2014).
[8] Twitter Inc.: Who's on Twitter, available from ⟨https://biz.twitter.com/ja/whos-twitter⟩ (accessed 2015-09).
[9] Facebook, Inc.: Facebook Reports First Quarter 2014 Results, available from ⟨http://investor.fb.com/releasedetail.cfm?ReleaseID=842071⟩ (accessed 2015-09).
[10] Fotiou, N., Nikander, P., Trossen, D. and Polyzos, G.C.: Developing information networking further: From PSIRP to PURSUIT, *Broadband Communications, Networks, and Systems*, Vol.66, Springer Berlin Heidelberg (2012).
[11] Wackerly, S.: OpenFlow Hybrid Mode, Open Daylight Developer Design Forum (Sep. 2014), available from ⟨https://wiki.opendaylight.org/images/1/1d/ODL_Hybrid_Mode.pdf⟩ (accessed 2015-01).
[12] OPEN NETWORK FOUNDATION, OpenFlow Switch Specification Version 1.3.3 (Sep. 2013), available from ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.3.pdf⟩ (accessed 2015-01).
[13] Akiyama, T. et al.: Proposal for a New Generation SDN-Aware Pub/Sub Environment, *Proc. 13th International Conference on Networks* (*ICN 2014*) (2014).
[14] Akiyama, T. et al.: SAPS: Software Defined Network Aware Pub/Sub – A Design of the Hybrid Architecture utilizing Distributed and Centralized Multicast, *IEEE 39th Annual International Computers, Software & Applications Conference* (*COMPSAC 2015*) (July 2015).
[15] Teranishi, Y.: PIAX: Toward a framework for sensor overlay network, *Proc. 6th Consumer Communications and Networking Conference* (*CCNC 2009*) (2009).
[16] Castro, M. et al.: 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).
[17] Bhattacharyya, S.: An Overview of Source-Specific Multicast (SSM), IETF RFC 3569 (July 2003).
[18] Baehni, S., Eugster, P.T. and Guerraoui, R.: Data-aware multicast, *Proc. International Conference on Dependable Systems and Networks 2004*, pp.233–242 (July 2004).
[19] Setty, V., Van Steen, M., Vitenberg, R. and Voulgaris, S.: PolderCast: Fast, Robust, and Scalable Architecture for P2P Topic-Based Pub/Sub, *Proc. 13th International Middleware Conference*, pp.271–291 (2012).
[20] Luby, M. et al.: The Use of Forward Error Correction (FEC) in Reliable Multicast, IETF RFC 3453 (Dec. 2002).
[21] Adamson, B. et al.: NACK-Oriented Reliable Multicast (NORM) Transport Protocol, IETF RFC 5740 (Nov. 2009).
[22] Boneh, D. and Franklin, M.: Identity-Based Encryption from the Weil Pairing, *Advances in Cryptology* (*CRYPTO 2001*), Lecture Notes in Computer Science, Vol.2139, No.2001, pp.213–229 (Aug. 2001).
[23] Zhang, X. et al.: Protocol Design for Scalable and Reliable Group Rekeying, *IEEE/ACM Trans. Networking* (*TON*), Vol.11, No.6 (Dec. 2003).
[24] Weis, B., Gross, G. and Ignjatic, D.: Multicast Extensions to the Security Architecture for the Internet Protocol, IETF RFC 5374 (Nov. 2008).
[25] Baugher, M., Rowles, S. and Hardjono, T.: The Group Domain of Interpretation, IETF RFC 6407 (Oct. 2011).
[26] Ishii, S. et al.: A study on designing OpenFlow controller RISE 3.0, *Proc. 19th IEEE International Conference on Networks* (*ICON*) (2013).
[27] Tootoonchian, A. and Ganjali, Y.: HyperFlow: A distributed control plane for OpenFlow, *Proc. Internet Network Management Conference on Research on Enterprise Networking* (*INM/WREN '10*), USENIX Association (2010).
[28] Watanabe, T., Omizo, T., Akiyama, T. and Iida, K.: ResilientFlow: Deployments of Distributed Control Channel Maintenance Modules to Recover SDN from Unexpected Failures, *IEEE Intn'l Conference on Design of Reliable Communication Networks* (*DRCN 2015*) (Mar. 2015).
[29] Al-Fares, M., Loukissas, A. and Vahdat, A.: A scalable, commodity data center network architecture, *ACM SIGCOMM Computer Communication Review*, Vol.38, No.4 (2008).
[30] Banno, R. 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, pp.105–116 (Mar. 2015).
[31] Teranishi, Y., Banno, R. and Akiyama, T.: Scalable and Locality-Aware Distributed Topic-based Pub/Sub Messaging for IoT, *Proc. 2015 IEEE Global Communications Conference* (*GLOBECOM 2015*): Selected Areas in Communications: P2P Networking (Dec. 2015).
[32] Zipf, G.: *Human Behaviour and the Principle of Least-Effort*, Addison-Wesley (1949).
[33] Page, C.: Twitter has almost 430 million inactive users, the *INQUIRER* (Apr. 2014), available from ⟨http://www.theinquirer.net/inquirer/news/2339684/twitter-has-almost-430-million-inactive-users⟩ (accessed 2016-02).
[34] Lindsay, B.R.: Social media and disasters: Current uses, future options, and policy considerations, *Congressional Research Service*, Vol.41987 (2011).

Toyokazu Akiyama received his B.E., M.E. and Ph.D. degrees in Information Systems Engineering from Osaka University, Japan in 1997, 1999 and 2003, respectively. Since 2000, he worked as a Research Associate (Assistant Professor) at the Cybermedia Center, Osaka University. Currently, he is an Associate Professor of the Faculty of Computer Science and Engineering, Kyoto Sangyo University. His research interests include distributed systems and internet applications. He is a member of the IEEE Computer Society, IEICE.

Yuuichi Teranishi received his M.E. and Ph.D. degrees from Osaka University, Japan, in 1995 and 2004, respectively. From 1995 to 2004, he was engaged Nippon Telegraph and Telephone Corporation (NTT). From 2005 to 2007, he was a Lecturer of Cybermedia Center, Osaka University. From 2007 to 2011, he was an Associate Professor of Graduate School of Information Science and Technology, Osaka University. Since August 2011, he has been a research manager and project manager of National Institute of Information and Communications Technology (NICT), Japan. His research interests include technologies for distributed network systems and applications. He is a member of IEEE.

Ryouhei 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 IEICE, and JSSST.

Katsuyoshi Iida received his B.E., M.E. and Ph.D. degrees in Computer Science and Systems Engineering from Kyushu Institute of Technology (KIT), Iizuka, Japan in 1996, in Information Science from Nara Institute of Science and Technology (NAIST), Ikoma, Japan in 1998, and in Computer Science and Systems Engineering from KIT in 2001, respectively. Currently, he is an Associate Professor in the Global Scientific Information and Computing Center, Tokyo Institute of Technology, Tokyo, Japan. His research interests include network systems engineering such as network architecture, performance evaluation, QoS, and mobile networks. He is a member of the WIDE project and IEEE. He received the 18th TELECOM System Technology Award, and Tokyo Tech young researcher's award in 2003, and 2010, respectively.

Yukiko Kawai is an Associate Professor at Faculty of Computer Science and Engineering, Kyoto Sangyo University. She received the M.S. and Ph.D. degrees in Information Science and Technology from Nara Institute of Science and Technology, in 1998 and 2001, respectively. She has worked as a research fellow at the National Institute of Information and Communications Technology from 2001 to 2006. Since 2006 she has been a lecturer at Kyoto Sangyo University. Her research interests include data mining, information analyzing and Web information retrieval. She is a member of DBSJ and IEICE.