Topic-based Allocation of Distributed Message Processors on Edge-Servers for Real-time Notification Service

Tomoya Tanaka
Graduate School of System Informatics
Kobe University
Kobe, Japan
tomoya.tanaka@fine.cs.kobe-u.ac.jp

Tomio Kamada
Graduate School of System Informatics
Kobe University
Kobe, Japan
kamada@fine.cs.kobe-u.ac.jp

Chikara Ohta
Graduate School of Science, Technology and Innovation
Kobe University
Kobe, Japan
chikara@m.ieice.org

***Abstract*—The importance of real-time notification has been growing for social services and Intelligent Transporting System (ITS). As an advanced version of Pub/Sub systems, publish-process-subscribe systems, where published messages are spooled and processed on edge servers, have been proposed to achieve data-driven intelligent notifications. In this paper, we present a system that allows a topic to be managed on multiple edge servers so that messages are processed near publishers and subscribers, even when publishers are spread over a wide area. However, permitting a topic to use multiple edge servers can cause exhaustion of resources as the number of topics and consumed resources by each topic increase. We formulate the delay caused by the depletion of edge server resources to find the optimal allocation of topics on limited edge server resources. As the optimization problem is NP-hard, we propose heuristics leveraging the given locality and the pub/sub relationships observed between clients to use the edge server resources efficiently while delivering real-time notifications. The experimental results demonstrate that the proposed method reduces the delay to deliver notifications and the effectiveness of the strategy exploiting the relationships between clients.**

***Index Terms*—Real-time notification, multi-access edge computing, publish-process-subscribe model**

I. Introduction

In recent years, the importance of real-time and data-driven notification has been growing dramatically for applications such as social services, IoT applicaitons, and Intelligent Transporting System (ITS) [1]. For example, real-time decision-making services proposed in ITS are expected to react immediately to changes in traffic conditions, analyze the current conditions, and provide optimal behavior for vehicles [2]. To achieve instant reactions to changed conditions and immediate data-driven notifications to clients, Multi-access Edge Computing (MEC) and Pub/Sub messaging models have been used.

MEC, proposed by the European Telecommunications Standards Institute (ETSI), enables ultra-low latency and high bandwidth by geographically distributing computation and storage resources to edge servers [3]. MEC has already been applied to IoT, AR / VR, and traffic management [1]. Pub/Sub model can be adapted to the design of large-scale distributed systems such as MEC [4]. Topic-based and content-based Pub/Sub model has been used in social interaction message notifications and event notifications among vehicles [5], [6].

In a more sophisticated version of Pub/Sub, spooling published messages on edge servers enables data-driven notifications supported by an analysis of the spooled messages, which is called publish-process-subscribe pattern [7].

We consider a publish-process-subscribe system, where each edge server generates and transmits the data-driven notifications near publishers and subscribers to achieve real-time notifications. Each edge server has a message processor for each topic managed by the edge server. The message processor functions as a message spooler and an analyzer of the spooled messages. Each edge server functions as a broker which receives messages from publishers and delivers the notifications produced by the message processors to the clients. Our system generates notifications and delivers them to subscribers in the following steps. When an edge server receives a message, it transmits this message to the message processor of the matching topic. The message processor generates notifications by analyzing the published message with the spooled messages, and returns the produced notifications to the edge server. Finally, the edge server disseminates the produced notifications to the subscribers.

In our system, a topic can be managed on multiple edge servers so that all publishers to a topic can send messages to their nearby edge server, considering applications where publishers to a topic spread over a wide area [8]. As an example of such applications, let us consider preventing accidents caused by a speeding vehicle. In this case, publishers such as IoT sensors and cameras in a wide area spanning several intersections, would send messages to a topic. Then, edge servers should send alert notifications to subscribers in the immediate vicinity of the speeding vehicle as well as subscribers further away.

Our system enables real-time notifications regardless of the locality of publishers and subscribers with the following advantages: (1) publishers can send messages immediately to their nearby edge server (e.g. the nearest one) which manages their target topic; and (2) subscribers can receive notifications

immediately from their nearby edge server. On the whole, each edge server that manages a topic can mediate between publishers and subscribers for each area according to the position of the edge server, cooperating with other servers. Note that a message published to a message processor on an edge server is also added to the message processors of the other edge servers. Thereby, each edge server can use messages published to other edge servers.

Due to the limited resources on edge servers, preparing a message processor on multiple edge servers for a topic causes exhaustion of the storage capacity. An efficient use of computational resources represented by brokers, which perform streaming message analysis in the publish-process-subscribe paradigm, is presented in [9]. In contrast, our focus is to utilize storage capacity on edge servers efficiently while enabling a topic to be managed on multiple edge servers.

In this paper, we propose an adequate allocation of topics on edge servers with limited edge server resources to achieve real-time notifications in the publish-process-subscribe system. A simple idea to resolve storage capacity over-consumption caused by preparing a topic's message processor on multiple edge servers consists in decreasing the number of edge servers to manage the topic. Though this measure reduces storage capacity usage, it spoils proximity between several publishers/subscribers and message processors. Therefore, decreasing the number of edge servers to manage a topic causes delays due to the marred proximity. Because of the involved trade-offs, investigating the optimal allocation of topics is necessary to enable real-time notifications.

This paper makes the following key contributions:

- We formulate the delay to deliver notifications to subscribers considering the resource constraints on edge servers as an optimization problem. The formulated problem gives optimal allocation of topics on edge servers in a publish-process-subscribe system.
- As the formulated optimization problem is NP-Hard, we propose a heuristics named RELOC, which allocates topics on edge servers so that notifications can be delivered to subscribers immediately. The method is constructed based on an analysis of the formulated optimization problem. We exploit given locality and topic-derived relationships observed among publishers and subscribers to use storage capacity efficiently while keeping proximity between clients and a message processor.

The remainder of this paper is organized as follows: Section II clarifies the position of our research, giving an overview of related work. In Section III, We formulate a cost model based on the presented notification system. In Section IV, we elaborate on our client assignment method with an analysis of the formulated cost model. Simulation results are shown to demonstrate the performance of our method in Section V.

II. Related Work

Publish-process-subscribe based real-time communication is investigated in several recent studies [7]. The message processing causes critical problems in achieving real-time notifications due to the processing load. Khare et al. present techniques to realize a scalable broker architecture that balances data publication and processing load for publish-process-subscribe systems operating at the edges, and ensures Quality-of-Service (QoS) on a per-topic basis [9]. In the publish-process-subscribe paradigm, we further consider applications where publishers spread over a wide area [8]. For such applications as well as ones where publishers and subscribers close together, managing a topic on multiple edge servers or brokers, which is not assumed in [9], has the potential to enable real-time notifications. However, prepared message processors on multiple servers could cause storage capacity over-consumption.

Considering the limited resources of edge servers, many researchers focus on proactively fetching content on edge servers to improve latency for clients to obtain content. The criteria to determine which content should be cached and where include: (1) content request probability [10], [11], where the most popular content is cached on edge servers; or (2) client mobility [12], [13], where content is cached on an edge server near the client who will request the content. In contrast, in publish-process-subscribe systems, the messages or data objects to be stored on edge servers is determined by topics and publishers/subscribers relationships. Nagato et al. propose a distributed data framework in a pub/sub paradigm. It enables proactive caching based on pub/sub relationships given by developers, but it does not consider resource capacity nor data processing [14]. In this research, we propose the efficient use of the limited storage capacity and other resources of edge servers with adequate allocation of topics to deliver real-time notifications. In [15], we are also investigating the trade-off between the network latency and storage consumption assuming no storage capacity limitation. To the best of our knowledge, topic-allocation-based messages management on edge servers in publish-process-subscribe paradigm has not been presented before.

III. System model and problem formulation

In this section, we investigate the optimal allocation of topics considering the resource constraints of edge servers to enable real-time notification in publish-process-subscribe system by formulating an optimization problem. The cost of the formulated optimization problem is as the delay between the publication of a message and the transfer of notifications to subscribers.

The optimization problem is formulated with the following assumptions: (1) Each publisher is assigned to an edge server which manages the publisher's target topic; (2) Each subscriber is also assigned to an edge server, but it only relays notifications to the subscribers without managing the subscriber's target topic; (3) The assigned edge server of each publisher manages the publisher's target topic, but when storage capacity on the server is exhausted, it prepares the message processor with spooled messages for the topic on a cloud server; and (4) When the assigned server does not have the message processor, published messages are forwarded to the cloud server.

We call the server assigned to each client the "home server". By assigning a home server to each client, publishers can send messages to their home server without having to locate an edge server which manages their target topic. On the other hand, subscribers can receive messages from their home server automatically, combined with push notification technology.

How we assign home servers to publishers determines the allocation of topics because the target topic of a publisher is always managed by its home server. If publishers for the same topic are assigned to different servers, the topic is managed by multiple edge servers. Moreover, we must be aware of home servers of subscribers to enable them to obtain notifications from their nearby servers. We define the assignment of a home server to each client as the strategy of the optimization problem. By defining the strategy as an assignment method for each client, we can consider the optimal allocation of topics from the point of view of publishers and subscribers. In the following sections, we show the relation between the cost and the strategy while modeling our publish-process-subscribe system followed by a formal definition of our optimization problem.

A. System model

We consider a field with L edge servers denoted $\mathcal{S} = \{s_1, s_2, \cdots, s_L\}$. In the field, M active clients, denoted $\mathcal{C} = \{c_1, c_2, \cdots, c_M\}$, move freely. Each client $c_m \in \mathcal{C}$ has a home server $s_l \in \mathcal{S}$. Let $\mathcal{C}_l$ denote the set of clients whose home server is s_l. The set of clients $\mathcal{C}$ is divided into L subsets, $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_L$. Note that the home server assignment of a client can change when the client moves.

We have assumed that when a publisher's home server does not have the message processor, published messages are forwarded to a cloud server. Let d^{C} denote the delay caused by the forwarding to a cloud server. We define that d^{C} does not depend on the server s_l from which the published message is forwarded. Delay in the path between a client c_m and its home server s_l is denoted as d_l^{H}. $d^{\mathrm{C}} > d_l^{\mathrm{H}}$ because of the expensive traffic through the core network. Note that we assume that resource exhaustion on edge servers does not cause additional delay compared to the propagation times when servers do not suffer from resource exhaustion. The propagation delay between edge servers is therefore kept at 0.

Fig. 1a shows the route and process from the publication of a message by a publisher c_m, to the delivery of a notification to a subscriber $c_{m'}$ when the message processor is present on the publisher's home server. The home server of publisher c_m is s_l, whereas the home server of subscriber $c_{m'}$ is $s_{l'}$. Fig. 1b presents the case where the message processor is not prepared on the publisher's home server. We divide the processes into two steps: before, and after the notification is generated. The delay in these two steps is denoted as D_m^{P} and $D_{m'}^{\mathrm{S}}$, respectively. In the case of Fig. 1a, D_m^{P} and $D_{m'}^{\mathrm{S}}$ equal d_l^{H} and $d_{l'}^{\mathrm{H}}$, respectively. On the other hand, in the case of Fig. 1b, D_m^{P} and $D_{m'}^{\mathrm{S}}$ equal $d_l^{\mathrm{H}} + d^{\mathrm{C}}$ and $d_{l'}^{\mathrm{H}} + d^{\mathrm{C}}$, respectively. Let r_l represent the probability that the publisher c_m's home server s_l does not have the message processor for a target topic. In this case, notifications are generated and transmitted from cloud server, which yields an additional delay d^{C} to both D_m^{P} and $D_{m'}^{\mathrm{S}}$.

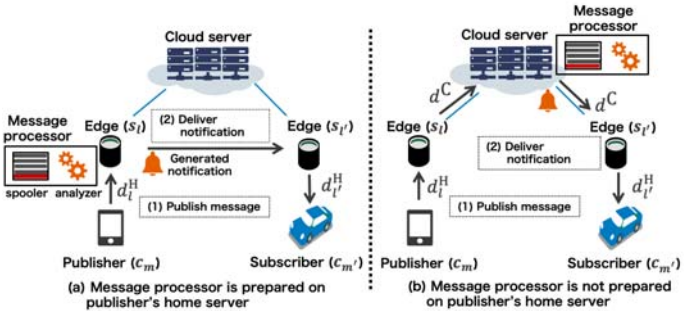

Fig. 1. Route and process to deliver a notification to a subscriber

B. Problem formulation

The strategy of the optimization problem is denoted by a binary matrix $\boldsymbol{X} = \{x_{m,l} | c_m \in \mathcal{C}, s_l \in \mathcal{S}\}$. If c_m's home server is s_l (*i.e.* $c_m \in \mathcal{C}_l$), $x_{m,l} = 1$, whereas $x_{m,l} = 0$ in the other cases. Our objective is to find the optimal $\boldsymbol{X}$ such that the delay $D_m^{\mathrm{P}} + D_{m'}^{\mathrm{S}}$ is minimized.

d_l^{H} involves delay caused by notable resource constraints on an edge server s_l. Let $d_{l,1}^{\mathrm{H}}$ denote the delay caused by the number of clients assigned to an edge server s_l exceeding the limit B. Moreover, let $d_{l,2}^{\mathrm{H}}$ denote the delay caused by the geographical distance between a client c_m and its home server s_l. The distance between c_m and s_l is denoted as $g_{m,l}$. We express d_l^{H} by the sum of $d_{l,1}^{\mathrm{H}}$ and $d_{l,2}^{\mathrm{H}}$, *i.e.* $d_l^{\mathrm{H}} = d_{l,1}^{\mathrm{H}} + d_{l,2}^{\mathrm{H}}$.

Reducing the value of r_l and keeping d_l^{H} small is incompatible. The value of r_l is kept small by decreasing the number of edge servers to manage a topic. In other words, the value of r_l is reduced by assigning publishers to fewer edge servers. However, assigning publishers to a smaller number of edge servers increases the value of $d_{l,1}^{\mathrm{H}}$ and $d_{l,2}^{\mathrm{H}}$ because it increases the number of clients assigned to an edge server and it decreases the proximity between clients and their home servers. We investigate an optimal strategy to resolve the trade-offs by formulating the problem mathematically.

We obtain an optimal solution based on the last N message publications. Let $\mathcal{P}_i$, and $\mathcal{S}_i$ denote the set of publisher, and subscribers who involve data transfer i ($1 \leq i \leq N$), respectively ($|\mathcal{P}_i| = 1, |\mathcal{S}_i| \geq 1$). We define the following optimization problem:

$$\min \quad \sum_{i=1}^{N} D_i, \tag{1}$$

$$\text{s.t.} \quad x_{m,l} = \{0, 1\}, \forall c_m \in \mathcal{C}, s_l \in \mathcal{S}, \tag{2}$$

$$\sum_{l=1}^{L} x_{m,l} = 1, c_m \in \mathcal{C}, \tag{3}$$

where

$$D_i = \sum_{c_m \in \mathcal{P}_i} D_m^{\mathrm{P}} + \frac{1}{|\mathcal{S}_i|} \sum_{c_m \in \mathcal{S}_i} D_m^{\mathrm{S}}. \tag{4}$$

D_i^{P} and D_i^{S} are expressed as

$$D_m^{\mathrm{P}} = D_m^{\mathrm{S}} = \sum_{l=1}^{L} (d_l^{\mathrm{H}} + r_l d^{\mathrm{C}}) x_{lm}, \tag{5}$$

where the probability that notifications are generated and transmitted at a cloud server r_l is expressed as

$$r_l = \boldsymbol{R}(1 - \frac{A}{u_l}), \tag{6}$$

where $\boldsymbol{R}(x)$ is a ramp function defined as $\boldsymbol{R}(x) = max(x, 0)$, A is the storage capacity on each edge server, and u_l is the total size of spooled messages managed by the edge server s_l. The messages are spooled in a message processor on the server s_l or in a message processor on a cloud server. r_l is 0 when the total size of spooled messages managed by server s_l is smaller than A. d_l^{H} involves delay $d_{l,1}^{\mathrm{H}}$ and $d_{l,2}^{\mathrm{H}}$ ($d_l^{\mathrm{H}} = d_{l,1}^{\mathrm{H}} + d_{l,2}^{\mathrm{H}}$). We express $d_{l,1}^{\mathrm{H}}$ as

$$d_{l,1}^{\mathrm{H}} = \beta \boldsymbol{R}(|\mathcal{C}_l| - B), \tag{7}$$

where β is an additional delay per an exceeded assignment of clients. We define $d_{l,2}^{\mathrm{H}}$ as

$$d_{l,2}^{\mathrm{H}} = \gamma g_{m,l}, \tag{8}$$

where γ is the delay per kilometer. The optimization problem is NP-Hard because it is a kind of set partitioning problem [16]. Therefore, we develop heuristics by an analysis of the formulated optimization problem.

IV. RELOC ALGORITHM

We develop a heuristics named RELOC (RElation and LOcality conscious Cooperative client assignment), which determines the strategy $\boldsymbol{X}$ that reduces the value of D_i resolving the trade-off between r_l and d_l^{H}. RELOC exploits notable features of publishers and subscribers, namely locality and topic-derived relationships.

Initially, RELOC divides a field into K clusters based on the position of edge servers using the K-means clustering algorithm. A client is only assigned to a server belonging to the closest cluster so that the value of $g_{m,l}$ is reduced. On top of restricting the number of potential home server assignments for each client, this prevents the concentration of clients on an edge server and reduces the variance of $|\mathcal{C}_l|$ in Equation 7. The number of clusters K is given in advance according to the strength of locality and mobility of clients. For example, in an application where publishers and subscribers are close together, we set K larger to more finely divide the field.

Secondly, RELOC assigns publishers who have strong relationships to the same edge server so that the number of edge servers which handles a given topic is reduced. The relationships among publishers are obtained by the following. We prepare a list of publishers for each topic and the matrix $\boldsymbol{Z} = \{z_{m,m'} | c_m, c_{m'} \in \mathcal{C}\}$, where $z_{m,m'}$ represents the number of times clients c_m and client $c_{m'}$ published to the same topic. $\boldsymbol{Z}$ is initialized as $\boldsymbol{Z} = \boldsymbol{0}$. If c_m and $c_{m'}$ are publishers to the same topic, $z_{m,m'}$ is incremented. After the preparation, RELOC obtains the set of publishers who have the strongest relationship with each publisher by adapting the matrix factorization described in [17] to $\boldsymbol{Z}$.

Publishers in different clusters are not assigned to the same server even if they have a strong relationship. Nevertheless, publishers to a topic are likely to be close together. Besides, the size of the clusters can be adjusted by changing the number of clusters created. Thus, RELOC can distribute clients to edge servers evenly, and keep the proximity between clients and their home servers by virtue of the restriction given by clustering, without breaking the relationships between publishers. We plan to determine the number of clusters to create according to the application features in future work.

Finally, RELOC takes account of the current resource usage of each edge server, which promotes cooperative management of messages between edge servers. By this operation, edge servers that are not heavily used are going to manage more topics. The overall operation of RELOC is presented in Algorithm 1.

Algorithm 1 RELOC: home server assignment to a client

Input: Client c_m to whom RELOC assign home server; the latest status of edge server s_l, namely r_l, $d_{l,1}^{\mathrm{H}}$, $d_{l,2}^{\mathrm{H}}$, which are defined as Equation 6, 7, 8, respectively; the number of clusters K; the number of extracted clients M'

Output: assigned home server's index l

```
1:  Divide field to K clusters.
2:  Choose M' clients who have the strongest relation with c_m
3:  C' ← set of indexes of chosen clients
4:  l ← index of a server with the smallest r_l, which belong to the same cluster as c_m
5:  for m' in C' do
6:      if c_m''s home server belongs to the same cluster as c_m's home server then
7:          if r_l = 0 and d^H_{l,1} = 0 then
8:              l ← index of c_m''s home server
9:              break
10:         end if
11:     end if
12: end for
```

1) *locality conscious:* In Line 1, the field with numerous edge servers is divided into K clusters using the K-means clustering algorithm.
2) *relation conscious:* From Line 2 to 8, except for Line 7, M' clients who have the strongest relationship with c_m is extracted based on matrix factorization technique. Subscripts of extracted clients are stored in $\mathcal{C}'$. A home server l is temporarily selected from servers which belong to the same cluster as c_m, and with the smallest r_l. If extracted clients include a client who belongs to the same cluster as c_m, l is reset to index of the extracted user's home server.
3) *Cooperation:* Line 7 is the additional condition to reset l. If resources on the extracted client's home server are exhausted ($r_l > 0 \vee d_{l,1}^{\mathrm{H}} > 0$), l is not updated.

TABLE I
PARAMETERS FOR THE SIMULATIONS

Parameter	Value	Parameter	value
L	100	d^{C}	5 ms
A	160 MB	B	100
β	1	γ	0.1
K	10	M'	100

TABLE II
CORRESPONDING M AND N TO REQUIREMENT RATIO (RR)

RR	M	N	RR	M	N
0.07	384	142	0.52	1865	914
0.81	2 714	1 361	0.95	3 123	1 587
1.28	4 058	2 053			

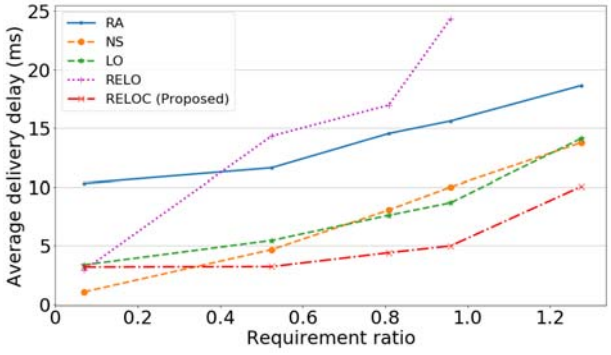

Fig. 2. Comparing average delivery delay Y of RA, NS, LO, RELO, and RELOC

V. PERFORMANCE EVALUATION

In this section, we conduct simulations to demonstrate the performance of RELOC. We adopt a metric Y as follows:

$$Y = \frac{1}{N}\sum_{i=1}^{N} D_i, \tag{9}$$

which is the average delay of the formulated cost defined as Equation 1. We also prepare additional metrics to identify bottlenecks, namely Y_1, Y_2, and Y_3 which denote the average of r_l, standard deviation of $|\mathcal{C}_l|$, and average of $g_{m,l}$, respectively.

$$Y_1 = \frac{1}{L}\sum_{l=1}^{L} r_l \tag{10}$$

$$Y_2 = \sqrt{\frac{1}{L-1}\sum_{l=1}^{L}(|\mathcal{C}_l| - \frac{1}{L}\sum_{l'=1}^{L}|\mathcal{C}_{l'}|)^2} \tag{11}$$

$$Y_3 = \frac{1}{M}\sum_{l=1}^{L}\sum_{m=1, c_m\in\mathcal{C}_l}^{M} g_{m,l} \tag{12}$$

We compare the performance of RELOC with 4 assignment methods, namely random assignment (RA), the nearest server assignment (NS), location conscious assignment (LO), and relation and location conscious assignment (RELO). RA assigns home servers to clients randomly, while NS assigns the closest server to clients, which is assumed in [14]. LO and RELO enable analyzing the effect by each element of the proposed method. LO is RELOC without relation consciousness and cooperation, while RELO is RELOC without cooperation. A list of parameters for the simulations is shown in Table I.

We obtain list of publishers from a real-world social network dataset (*user_sns.txt*) provided by Tencent Inc. [18]. We group `Followee-userid` by `Follower-userid` and make the arranged dataset, where each line includes a follower and her/his followees. We extract N lines from the arranged dataset and assign a topic to each line. Besides, we regard a follower and followees in each line as a list of clients who can take a role of publishers.

As preparation for the simulations, we distribute M clients that appeared in the N lines extracted from the dataset in a field of 10 km square field. We place the clients following a Gaussian distribution with a standard deviation of 2 km centered on a randomly chosen point in the field.

A home server is assigned to each client following the various assignment methods, including RELOC. The strategy $\boldsymbol{X}$ and d_l^{H} is fixed at this point. The home server of each of the M clients manages the topic to which the client publishes messages. Each message processor prepared for a topic holds 1 MB of spooled messages. Therefore, $\frac{A}{u_l}$ in Equation 6 is calculated as the number of topics managed by edge server s_l, multiplied by 1 MB. At this point, the value of r_l is fixed. The status of an edge server s_l, namely r_l, d_l^{H}, and the strategy $\boldsymbol{X}$ would be different depending on the method chosen to assign home servers. $\mathcal{P}_i$ and $\mathcal{S}_i$ are selected from clients in line i of the arranged dataset.

After the above preparation, We conduct N notification deliveries starting at $i = 1$ in the following steps: (1) $c_m \in \mathcal{P}_i$ publishes a message to its home server which manages the target topic; (2) The message is processed with the 1 MB of spooled messages and the notifications are generated; (3) produced notifications are delivered to the subscribers $c_m \in \mathcal{S}_i$. We conduct 5 trials for each data. In this simulation, we fixed the size of the spooled messages and the value of r_l, d_l^{H} remains constant. This assumes a situation where old messages are expired to utilize the limited resource of edge servers.

Fig. 2 shows the average delivery delay Y against the requirement ratio. The requirement ratio is defined as the ratio of maximum storage capacity that would be used, to $A \cdot L$, which is the total size of the storage capacity on edge servers. Corresponding M and N for each requirement ratio (RR) in our simulations are presented in Table II. From Fig. 2, we can observe that RELOC has the smallest delay to deliver notifications to clients. Even when maximum storage capacity that would be used is close to total size of storage capacity on edge servers (*i.e.* requirement ratio is close to 1.0), RELOC allocates topics on edge servers efficiently, and does not cause so much resource exhaustion which gives a delay of more than 6 ms.

The effectiveness of social consciousness, locality consciousness, and corporation in RELOC can be observed in

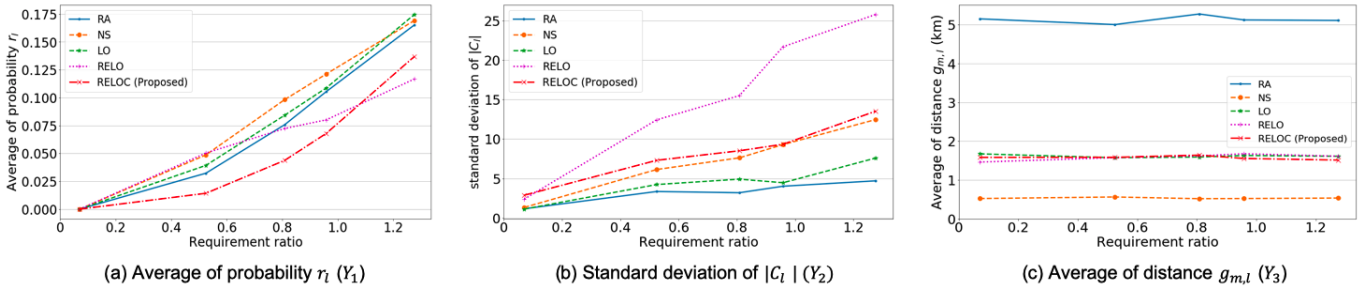

Fig. 3. Comparing the average of probability that a message processor managed by edge server s_l is allocated on a cloud server (Y_1), the standard deviation of the number of assigned clients to edge server s_l (Y_2), and the average of geographical distance between c_m and its home server s_l (Y_3) of RA, NS, LO, RELO, and RELOC

Fig. 3. The effect of social consciousness is demonstrated by the performance gains of RELOC and RELO over LO under high requirement ratio in Fig. 3a. In Fig. 3b, we can observe RELOC without corporation, (*i.e.* RELO), concentrates users to several servers as the requirement ratio increases. Thus, the effectiveness of cooperation is demonstrated. The advantage of LO, RELO, and RELOC over RA in Fig. 3c demonstrates the effectiveness of locality consciousness.

VI. Conclusion

In this paper, we presented a publish-process-subscribe system that allows published messages to a topic to be immediately processed by distributed message processors on edge servers. We formulated the delay caused by excessive edge server resource use by the message processors to find the optimized allocation of message processors on edge servers. Numerical experiments show that our heuristics constructed with an analysis of the formulated optimization problem give an efficient use of edge server resources, and reduce the delay. Future work involves comparing results by the optimal solution and heuristic solution, considering how to determine K adequately in Algorithm 1 exploiting several features of application users such as locality and mobility, and investigating time complexity and execution frequency to confirm that RELOC can be deployed to real environments.

Acknowledgement

We would like to thank Patrick Finnerty for his precious comments. This work was supported by JSPS KAKENHI Grant Number JP18H03232, JP20K11841. This work is partly carried out on StarBED which is provided by National Institute of Information and Communications Technology (NICT).

References

[1] A. Reznik, L. M. C. Murillo, Y. Fang, W. Featherstone, M. Filippou, F. Fontes, F. Giust, Q, Huang, A. Li, C. Turyagyenda, D. Wehner and Z. Zheng, "Cloud RAN and MEC: A perfect paring," *ETSI White Paper*, No. 23, 2018.

[2] S. Shaheen and R. Finson, "Intelligent transportation systems, " *Encyclopedia of Energy*, 2013.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher and V. Young, "Mobile edge computing a key technology towards 5G," *ETSI White paper*, No. 11, 2015.

[4] V. Setty, G. Kreitz, G. Urdaneta, R. Vitenberg and M. van Steen, "Maximizing the number of satisfied subscribers in pub/sub systems under capacity constraints," *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 2580-2588.

[5] N. Apolónia, S. Antaris, S. Girdzijauskas, G. Pallis and M. Dikaiakos, "SELECT: A distributed publish/subscribe notification system for online social networks," *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018, pp. 970-979.

[6] F. Zhang, B. Jin, W. Zhuo, Z. Wang and L. Zhang, "A content-based publish/subscribe system for efficient event notification over vehicular ad hoc networks," *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, 2012, pp. 64-71.

[7] B. Krishnamachari and K. Wright, "The publish-process-subscribe paradigm for the internet of things," *USC ANRG Technical Report*, 2017.

[8] L. Hou, L. Lei and K. Zheng, "Design on publish/subscribe message dissemination for vehicular networks with mobile edge computing," *2017 IEEE Globecom Workshops (GC Wkshps)*, 2017, pp. 1-6.

[9] S. Khare et al., "Scalable edge computing for low latency data dissemination in topic-based publish/subscribe," *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 214-227.

[10] Q. Li, W. Shi, Y. Xiao, X. Ge and A. Pandharipande, "Content size-aware edge caching: A size-weighted popularity-based approach," *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 206-212.

[11] T. Hou, G. Feng, S. Qin and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1-6.

[12] V. A. Siris, X. Vasilakos and G. C. Polyzos, "Efficient proactive caching for supporting seamless mobility," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1-6.

[13] L. Hou, L. Lei, K. Zheng and X. Wang, "A Q -learning-based proactive caching strategy for non-safety related services in vehicular networks," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4512-4520, 2019.

[14] T. Nagato, T. Tsutano, T. Kamada, Y. Takaki and C. Ohta, "Distributed key-value storage for edge computing and its explicit data distribution method," in *IEICE Transactions on Communications*, Vol. E103.B, No. 1, pp. 20-31, 2020.

[15] T. Tanaka, T. Kamada, and C. Ohta, "Distributed topic management in publish-process-subscribe systems on edge-servers for real-time notification service," *IEICE Communications Express*, vol. advpub, 2020.

[16] M. Lewis, G. Kochenberger and B. Alidaee, "A new modeling and solution approach for the set-partitioning problem," in *Computers & Operations Research*, Vol.35, No. 3, pp.807-813, 2008.

[17] R. Sebastian, Y. Limin, M. Andrew and M. Benjamin M., "Relation extraction with matrix factorization and universal schemas," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp.74-84.

[18] Tencento Inc., "Predict which users (or information sources) one user might foloow in Tencent Weibo," *Kaggle KDD Cup 2012 Track 1*, 2012. [Online] Available : https://www.kaggle.com/c/kddcup2012-track1