PAPER

# An Optimistic Synchronization Based Optimal Server Selection Scheme for Delay Sensitive Communication Services

**Akio KAWABATA**[†a)], ***Member***, **Bijoy Chand CHATTERJEE**[††], ***Nonmember***, ***and*** **Eiji OKI**[†††], ***Fellow***

**SUMMARY** In distributed processing for communication services, a proper server selection scheme is required to reduce delay by ensuring the event occurrence order. Although a conservative synchronization algorithm (CSA) has been used to achieve this goal, an optimistic synchronization algorithm (OSA) can be feasible for synchronizing distributed systems. In comparison with CSA, which reproduces events in occurrence order before processing applications, OSA can be feasible to realize low delay communication as the processing events arrive sequentially. This paper proposes an optimal server selection scheme that uses OSA for distributed processing systems to minimize end-to-end delay under the condition that maximum status holding time is limited. In other words, the end-to-end delay is minimized based on the allowed rollback time, which is given according to the application designing aspects and availability of computing resources. Numerical results indicate that the proposed scheme reduces the delay compared to the conventional scheme.

***key words:*** *real-time application, distributed processing, edge computing, optimistic synchronize algorithm, time warp, and server selection problem*

## 1. Introduction

Various network applications are being provided from a cloud environment with the development of virtualization technology and cloud services [1]. Besides, information and communication technology has been introduced into various industries, and many services utilizing the cloud environment in various industrial fields, such as agriculture, medical care, environment, and entertainment. In applications provided via a network, communication delay may cause degradation of service quality. The delay is recommended less than several tens of ms for applications that require real-time communication, such as telephone conferences [2], net-games [3], and music sessions [4] played with multiple locations. In addition, these applications need to process events in the actual occurrence order to ensure fairness among users.

Several study [5], [6] have been demonstrate the pros and cons of parallel and distributed systems for processing events in their actual occurrence order. Researchers are exploring their actual event occurrence while keeping the parallelism of processes. It is one of the significant issues in parallel and distributed systems [7], [8]. These research issues are typically handled by adapting different synchronization algorithms, which are mainly divided into two categories, namely conservative synchronization algorithm (CSA) and optimistic synchronization algorithm (OSA) [9]. CSA keeps the order of events to be their actual order by attaching the time information to the event. On the other hand, the OSA does not need to maintain the correct event order in advance. If the processing function receives a past event, it guarantees the order of events by rolling back the status and corrects its result. These approaches are applicable for delay-sensitive applications, such as network shooter games [10] or transaction management systems [11]. Since the processing history is continuously stored in the rollback approach, the memory consumption can be an issue, compared to CSA.

At the delay-sensitive communication services, the work in [12], [13] presented a server selection scheme for a distributed processing system. This work achieves a lower delay communication among multiple users while ensuring their actual event order at users compared to central processing or distributed processing with CSA. In [12], [13], the event order is reproduced in their actual order of occurrence based on CSA. As a result, CSA reproduces events in the occurrence order before processing applications, which introduces additional delay. OSA is typically used to realize better delay performance than CSA [14].

This paper proposes a server selection scheme based on OSA to minimize end-to-end delay under the condition that a maximum status holding time is limited. The maximum status holding time affects the application designing and the memory consumption of processing servers. Compared to CSA, the proposed scheme contributes to a lower delay for applications under the condition that the maximum delay of applications and maximum memory resources of servers are limited. This scheme is processed on a distributed processing system. An integer linear programming (ILP) problem is formulated to determine the optimal set of servers, each of which is assigned to users, for the proposed scheme. We prove that the decision version of the server selection problem in the proposed scheme is NP-complete.

We evaluate the proposed scheme in terms of delay and computation time for COST239 [15] and JPN [16] networks, and compare it to the conventional CSA-based server selection scheme. Numerical results indicate that the proposed scheme reduces the delay compared to the conventional scheme, by employing additional computations. Note

Manuscript received November 25, 2020.
Manuscript revised March 2, 2021.
Manuscript publicized April 9, 2021.

[†]The author is with the NTT Network Technology Laboratories, Musashino-shi, 180-8585 Japan.
[††]The author is with the South Asian University, New Delhi, India.
[†††]The author is with the Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: akio.kawabata.un@hco.ntt.co.jp

DOI: 10.1587/transcom.2020EBP3178

that the conventional scheme in [12], [13] is a server selection scheme based on CSA for distributed servers to minimize the end-end delay.

This paper is an extended version of [17] and the main additions are as follows. We present the related works on distributed systems for ensuring event occurrence order, low delay processing systems for network applications, and the optimization and approximate algorithms of server selection problems. As the server selection problem, the proposed scheme can set the maximum status holding time of each server as a constraint of server selection. This improvement is effective for the condition that the maximum memory resources of each server are different. We prove that the decision version of the server selection problem is NP-complete. We extensively evaluate delay characteristics depending on the given parameters, including the number of users, the maximum number of users who select the server, and the maximum status holding time. These evaluations contribute to confirming the effectiveness of the proposed scheme under various conditions.

The rest of the paper is organized as follows. Section 2 presents the related works on low delay communication and distributed systems. The prerequisite of server selection for the proposed scheme is presented in Sect. 3. The formulation of the proposed scheme is presented in Sect. 4. Section 5 presents the proof of NP-completeness of the server selection problem. In Sect. 6, we evaluate the proposed scheme in terms of delay and computation time compared to the conventional scheme. Finally, Sect. 7 concludes the paper.

## 2. Related Work

The edge computing technology [18] has been considered to reduce delay by executing applications on servers near the user. It is observed from the latest research that, when edge computing is incorporated, the performance is improved 20–70% compared to the standard web engine [19]. This technique is effective for server-client type applications, but it is not effective for multiple-user interactive communication processed at distributed servers, as synchronization on servers in the network causes extra delay.

For multiple-user interactive communication services, the works in [12], [13] presented server selection schemes based on CSA to reduce delay by ensuring the event occurrence order. These schemes contribute to making applications easily and provide a usage environment with low delay. The OSA-based approach, which processes events at arrival, is expected to achieve lower delay than the CSA-based approach. This is because OSA does not need to maintain the correct event order before application processing.

For ensuring the actual event occurrence order for distributed systems, the occurrence of event order in CSA is guaranteed consecutively by assigning the evidence of time to the event. In contrast, OSA does not require reproducing their actual occurrence order of events before processing. When a previous occurrence event arrives at the processing server, it guarantees the sequence of event occurrences by rolling back the application status and tuning its result. Time Warp [14] is known as a method for implementing OSA. At the Time Warp mechanism, each server processes events asynchronously with other servers based on optimistic synchronization. Memory consumption is one of the major constraints for the Time Warp. Another constraint is that there are events which cannot be rolled back once they are executed. The maximum time of rollback is introduced to manage these constraints, and this time is called global virtual time (GVT). In other words, it is not necessary to hold the status of events before GVT from the current time, and the processing can be completed since there is no need for the rollback.

The work in [20] introduced algorithms for reducing the number of statuses and memory consumption compared to Time Warp. Y. B. Lin et al. [21] presented an algorithm for selecting the checkpoint interval. The optimum checkpoint period is determined during the Time Warp execution; the status of cycles is saved at the checkpoint. These works contribute to reducing memory consumption in Time Warp. Our research work does not aim merely to reduce memory consumption but aims to minimize the communication delay using available memory resources.

Our research contributes to reducing the end-to-end delay for network applications in which each user interactively communicates at multiple locations. Our research is based on the OSA-based synchronization and achieves lower delay compared to the CSA-based approach. The delay advantage of OSA is that it allows processing events without waiting. Compared to the CSA-based approach, the end-to-end delay using OSA is smaller since there is no event waiting. However, the constraint to hold the status of events for the duration of GVT may restrict application designing and server memory resources. In our research, the delay is minimized considering the constraint of GVT, which is determined based on application designing and memory resources of servers. It is considered as a new approach to distributed processing that reduces delay with memory consumption as a constraint in OSA.

## 3. Prerequisite of Server Selection for Proposed Scheme

Figure 1 shows an example topology of users and servers. Each number attached to a link indicates a delay associated with that link. The user selects an optimal server from multiple distributed servers. All events are multi-casted to other distributed servers to synchronize the status of each distributed server.

### 3.1 Optimistic and Conservative Synchronizations

In the CSA-based scheme [12], [13], events are rearranged according to occurrence order before processing applications. In the OSA-based scheme, because of the rollback process, it is necessary to hold the past application status until the arrival of the slowest event. Figure 2 shows an example of synchronization based on OSA and compares it

with the CSA-based synchronization. We define a unit-less time for comparison. $S_n$ represents the status of application on each server, where $n$ is the status index. $S_n$ is updated $S_{n+1}$ by receiving an event. The event occurrence times of users 1, 2, and 3 are 0.00, 0.01, and 0.02, respectively. In Fig. 2(a), the status is rolled back 0.01 ago and changed from $S_1$ to $S_2$, when server B receives event 2. In Fig. 2(b), the status is changed from $S_1$ to $S_2$ after waiting for 0.03 until server B processes the event. The dotted line arrows drawn against $S_n$ indicate the rollback process in Fig. 2(a) and the event waiting process in Fig. 2(b). Since the statuses after receiving server B of events 2 and 1 are different, we describe them as different statuses with $S_2$ and $S_5$.

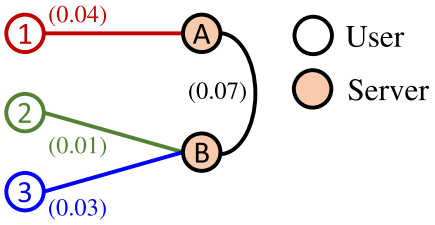
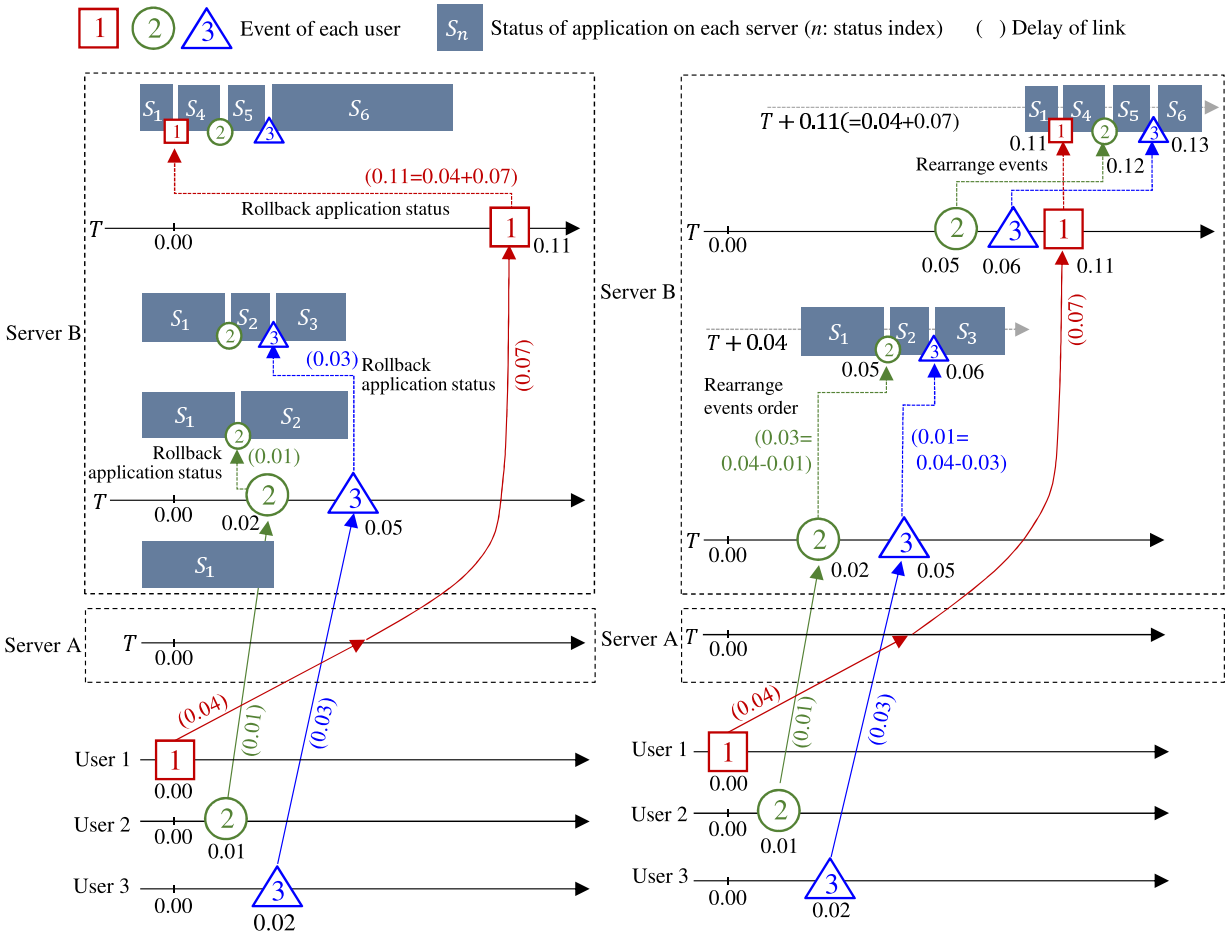



**Fig. 1** Example topology of users and servers. Each number attached to a link indicates a delay associated with the link.

In OSA adopted by the proposed scheme, the application status of server B rolls back 0.01 when the event of user 2 arrives, since the event of user 2 occurred at 0.01 time ago. Similarly, the application status of server B rolls back 0.03 when the event of user 3 arrives. After all, the application status at server B needs to be held until the slowest event arrival of user 1 via server A. Therefore, the maximum status holding time is 0.11 (= 0.04 + 0.07). The maximum delay using OSA is 0.08 (= 0.04 × 2); it is the round-trip time between user 1 and server A which is the maximum delay between a user and a server.

In CSA [12], [13], events from users are rearranged based on the maximum delay between a user and a server. The maximum delay between a user and a server is 0.04, which is the delay between user 1 and server A. In this case, each user event between user and server is processed after 0.04 from the actual time $T$. When the event of user 3 arrives at server B, the event waits for 0.01 (=0.04-0.03). This is because the maximum delay between a user and a server is 0.04 and that between user 3 and server B is




(a) Optimistic synchronization algorithm

(b) Conservative synchronization algorithm

**Fig. 2** Example of status synchronization using OSA and CSA.

**Table 1** Delay and status holding time of example networks.

| | Synchronization algorithm | |
|---|---|---|
| | Optimistic | Conservative |
| Maximum delay | 0.08 (= 0.04 × 2) | 0.15 (= 0.04 × 2 + 0.07) |
| Maximum status holding time | 0.11 (= 0.04+0.07) | - |

0.03. When the event of user 2 arrives at server B, the event waits for 0.03 (=0.04-0.01). In CSA, events from servers are rearranged based on the maximum delay between a pair of servers. The maximum delay between a pair of servers is 0.07, which is the delay between severs A and B. In this case, each user's event between a pair of servers is processed after 0.07 from $T + 0.04$. When the event of user 1 arrives at server B via server A, the events of users 2 and 3 additionally wait for 0.07. From this rearrangement, all events are reproduced at servers in the event occurrence order. After all, each event is reordered at each server with $T + 0.11$, where $T$ is an actual event time. The time of 0.11 is the sum of the maximum delay between a user and a server (0.04) and the maximum delay between a pair of servers (0.07). The maximum delay using CSA is 0.15 (= $0.04 \times 2 + 0.07$).

Table 1 shows the maximum delays and the maximum status holding times using OSA and CSA. From Table 1, it is observed that OSA has better delay performance than CSA, and hence the proposed scheme adopts OSA.

### 3.2 Necessity of Server Selection Problem

In OSA, if a user can connect to multiple servers, the maximum end-to-end delay varies, since the delay between the user and the server differs depending on the selected server. The end-to-end delay is considered as a round-trip time between a user and the selected server. When a server connected with users is determined to minimize the end-to-end delay, the constraint of the maximum status holding time for application designing and memory resources of servers must be considered. The waiting time at servers when the farthest user's event arrives must not exceed the maximum status holding time. Let $L_{\mathrm{APL}}$ denote the maximum status holding time of an application, and $L_i$ denote the maximum status holding time of server $i$. $L_{\mathrm{APL}}$ and $L_i$ are given in advance by considering the application designing and memory consumption at each server, respectively. Figure 3 shows an example of two types of server selection in the same network.

The delays using OSA with selections 1 and 2 in Figs. 3(a) and (b) are 0.08 (= 0.04×2) and 0.14 (= 0.07×2), respectively. The maximum status holding times using Figs. 3(a) and (b) are 0.14 (= 0.04 + 0.10) and 0.12 (= 0.07+0.05), respectively. In Fig. 3(a), user 1 selects the nearest server, but the maximum status holding time is greater than Fig. 3(b). If $\min(L_{\mathrm{APL}}, L_i) \geq 0.14$, selection 1 is better than selection 2; $i$ used in $L_i$ is applied for all servers in Figs. 3. If $0.12 \leq \min(L_{\mathrm{APL}}, L_i) < 0.14$, only selection 2 is feasible; selection 1 is infeasible. If $\min(L_{\mathrm{APL}}, L_i) < 0.12$,

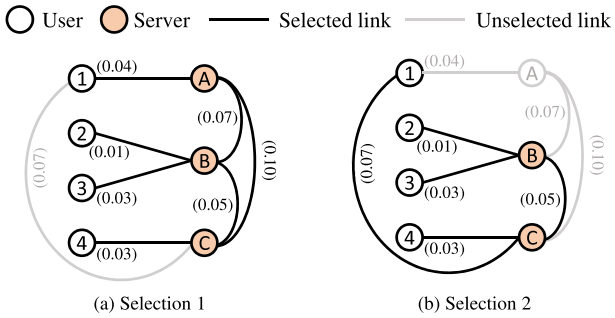



**Fig. 3** Example of server selections considering OSA. Each number attached to a link indicates a delay associated with the link.

no feasible solution is allowed. Thus, it is necessary to select the optimal set of servers for minimizing the end-to-end delay in the proposed scheme under the condition that the constraint of maximum status holding time is satisfied.

Note that the server selection problem assigns each user to a suitable server, and hence after the server selection problem, an optimal set of servers is determined, which minimizes the end-to-end delay. The sever selection problem for the proposed scheme is formulated in the following section.

## 4. Formulation of Sever Selection Problem for Proposed Scheme

The network is modeled as undirected graph $G(V, E)$, where $V$ and $E$ indicate sets of nodes and non-directional links, respectively. The set of users is represented by $V_{\mathrm{U}} \subseteq V$. The set of servers is represented by $V_{\mathrm{S}} \subseteq V$. $V_{\mathrm{U}} \cup V_{\mathrm{S}} = V$ since a node is either a user or a server, and $V_{\mathrm{U}} \cap V_{\mathrm{S}} = \emptyset$ since there is no node that is both a user and a server. The set of links between user and server is represented by $E_{\mathrm{US}} \subseteq E$; a link between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$ is symbolized by $(p, i) \in E_{\mathrm{US}}$. The set of links between server and server is represented by $E_{\mathrm{SS}} \subseteq E$; a link between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}}$ is symbolized by $(i, j) \in E_{\mathrm{SS}}$. $E_{\mathrm{US}} \cup E_{\mathrm{SS}} = E$ since a link is either a link between user and server or a link between server and server, and $E_{\mathrm{US}} \cap E_{\mathrm{SS}} = \emptyset$ since there is no link that is both a link between user and server and a link between server and server. The delay of link $(p, i) \in E_{\mathrm{US}}$ is represented by $d_{pi}$. The delay of link $(i, j) \in E_{\mathrm{SS}}$ is denoted by $d_{ij}$.

The maximum number of users that select server $i \in V_{\mathrm{S}}$ is denoted by $M_i$. The maximum number of selected servers is denoted by $Y_{\mathrm{MAX}}$. $z_{kl}$ is a binary variable for $(k, l) \in E$, where $z_{kl} = 1$ if link $(k, l)$ is selected, and $z_{kl} = 0$ otherwise. $y_i$ is a binary variable for $i \in V_{\mathrm{S}}$, where $y_i = 1$ if sever $i$ is selected, and $y_i = 0$ otherwise. The maximum value of $d_{pi}$, $(p, i) \in E_{\mathrm{US}}$, over selected links is represented by $D_{\mathrm{US}}^{\max}$.

The server selection problem in the proposed scheme is formulated as an ILP problem by:

$$\text{Objective min} \quad 2D_{\mathrm{US}}^{\max} \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in V_{\mathrm{S}}} z_{pi} = 1, \forall p \in V_{\mathrm{U}} \tag{1b}$$

$$\sum_{p \in V_U} z_{pi} \leq M_i, \forall i \in V_S \quad (1c)$$

$$z_{pi} \leq y_i, \forall p \in V_U, i \in V_S \quad (1d)$$

$$d_{pi} z_{pi} \leq D_{US}^{max}, \forall (p,i) \in E_{US} \quad (1e)$$

$$d_{pi} z_{pi} + d_{ij} z_{ij} \leq \min(L_{APL}, L_i), \quad \forall i \in V_S, (p,i) \in E_{US}, (i,j) \in E_{SS} \quad (1f)$$

$$y_i + y_j - 1 \leq z_{ij}, \forall (i,j) \in E_{SS} \quad (1g)$$

$$z_{ij} \leq y_i, \forall i \in V_S, (i.j) \in E_{SS} \quad (1h)$$

$$z_{ij} \leq y_j, \forall j \in V_S, (i,j) \in E_{SS} \quad (1i)$$

$$\sum_{i \in V_S} y_i \leq Y_{MAX} \quad (1j)$$

$$z_{kl} \in \{0,1\}, \forall (k,l) \in E \quad (1k)$$

$$y_i \in \{0,1\}, \forall i \in V_S. \quad (1l)$$

Equation (1a) expresses the objective function, which minimizes the delay. The delay is obtained by considering the round-trip time of the maximum value of selected links among all users and all servers. Equation (1b) represents that one link is selected between user and server. Equation (1c) represents that the sum of the number of users who select server $i \in V_S$ does not exceed $M_i$. Equation (1d) indicates that $z_{pi} \leq y_i$ represents that server $i$ is selected, $y_i = 1$, if it is selected by at least one user. Equation (1e) expresses that $D_{US}^{max}$ is the maximum value of $d_{pi}$, $(p,i) \in E_{US}$, for all selected links with (1a). Equation (1f) expresses that the maximum status holding time must not exceed $\min(L_{APL}, L_i)$, $\forall i \in V_S$. In other words, Eq. (1f) expresses that the smaller of the time due to maximum server memory resources and the time due to permissible applications delay is the constraint value as the maximum status holding time. Equations (1g)–(1i) indicate that $y_i \times y_j = z_{ij}$ is satisfied at sever $i \in V_S$, server $j \in V_S$, and link $(i,j) \in E_{SS}$. Equation (1j) represents that the sum of selected servers does not exceed $Y_{MAX}$. Equations (1k)–(1l) state that $z_{kl}$ and $y_i$ are binary variables.

The decision variables are $D_{US}^{max}$, $z_{kl}$, and $y_i$. The given parameters are $L_{APL}$, $L_i$, $d_{pi}$, $d_{ij}$, $M_i$, and $Y_{MAX}$.

## 5. Time Complexity Analysis for OSA-Based Server Selection

We prove that the OSA-based server selection decision problem (OSAD) is NP-complete as follows.

**Definition 1:** Given sets of $P$ users and $N$ servers, the distances between server-user pairs, the distances between server-server pairs, the maximum number of users for each server $M_i$, and the maximum status holding time $\min(L_{APL}, L_i)$. Is it feasible to arrange an assignment such that each user is allotted to an appropriate server?

**Theorem 1:** The OSAD problem is NP-complete.

**Proof 1:** OSAD is in NP, as we can verify whether a given solution of OSAD is feasible in polynomial time. We verify if each user is connected to a server in $O(P)$. We verify, in $O(PN)$, if the constraint of maximum status holding time is gratified for all possible combinations of each user and each pair of the server connected with the user and another server connected with another user. The overall time complexity for this verification is $O(PN)$.

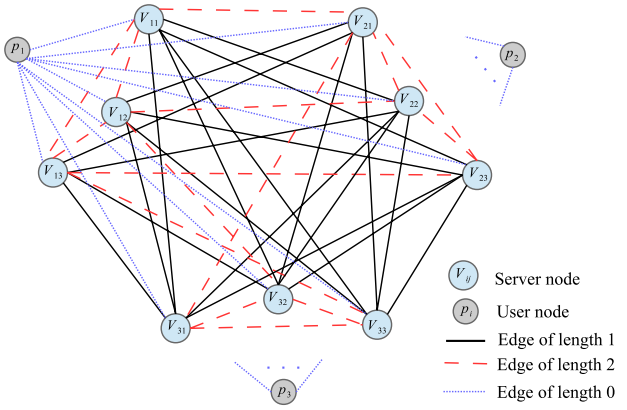



**Fig. 4** Graph $G'$ that corresponds to 3-SAT problem that has three clauses.

We consider a well-known NP-complete problem, which is the 3-SAT problem [22]. To prove that the OSAD problem is NP-complete, the 3-SAT problem is polynomial time reducible to the OSAD problem. We define the 3-SAT problem as follow:

Given a set of $n$ variables that are Boolean, and $k$ clauses, each of which has three elements. Can it be possible to make a truth assignment in such way that all clauses are satisfied?

An instance of the OSAD problem is constructed from any instance of the 3-SAT problem in the similar way of [12]; Fig. 4 depicts the schematic diagram of the construction.

- A graph $G'$ is created with $3k$ servers collected into $k$ sets of three nodes $V_{ij}$, where $i = 1, 2, \cdots, k$ and $j = 1, 2, 3$, and $k$ user nodes.
- Each server is linked with an edge
  - One is assigned to set the edge length $(V_{ij}, V_{i'j'})$ when $i \neq i'$. $V_{ij}$ and $V_{i'j'}$ are not negations to each other, which means that the edge represents two nodes corresponding to elements that have a compatible true assignment.
  - Otherwise, two is assigned to set the edge length.
- Each user is linked to all servers with edges that have length zero.

We set the capacity of each server $M_i$ to one to complete the OSAD instance. We set min $(L_{APL}, L_i)$=1, $\forall i \in V_S$.

The OSAD instance is feasible if and only if there exists a true 3-SAT assignment, which is shown in the following.

Assume that there exists a true 3-SAT assignment. Thereafter, $k$ nodes can be selected, one corresponds to a true assignment from each clause, which are all linked in $G'$ with edge length one. The $k$ users are assigned to the $k$ selected servers, one each. Since no edge with length greater than one in $G'$ is used, the maximum status holding time

constraint is satisfied. Thus, the OSAD instance is feasible.

Contrariwise, let assume that the OSAD instance is feasible. Then, there exists a set of $k$ fully linked servers with edge length of at most one between them. According to definition of graph $G'$, these nodes correspond to variables with compatible true assignment. Thus, the truth assignment that sets the variable corresponding to the $k$ nodes to true satisfies all the clauses. Therefore, the 3-SAT instance is feasible. ■

The optimal server selection problem (OSSP) presented in section 4 is NP-hard, since OSAD is proved to be NP-complete. No polynomial-time algorithm to an NP-hard problem has been found so far. It is reasonable to solve OSSP by using a solver that handles ILP in case that the computation time is acceptable.

## 6. Performance Evaluation

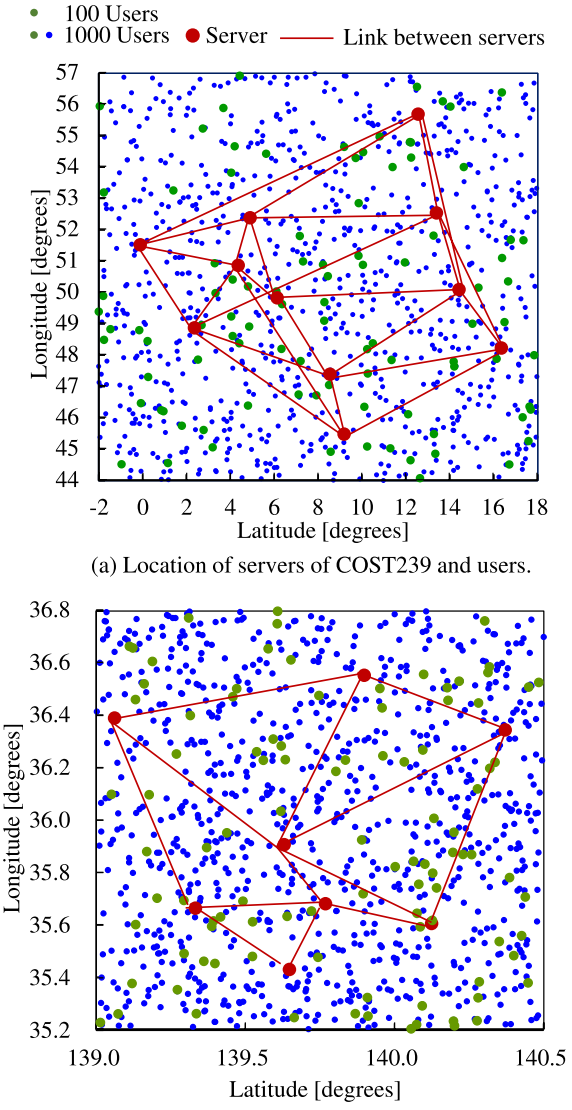
We assess the performance of the proposed scheme in terms of delay compared to the conventional scheme [12]. In advance, we estimate delays between server-to-server and user-to-server. The delay for processing at the server is ignored. Each user can select any server. We solve the ILP problem in (1a)–(1l) using Solving Constraint Integer Programs (SCIP) [23]; the computer is configured with Intel(R) Core(TM) i7-9750 2.60 GHz 12 core, and 16 GB memory is used.

We evaluate the proposed scheme with two different topology types to observe the topology dependency of the proposed scheme. We assume that the servers are located at the nodes of COST239 [15] and JPN Kanto region [16], as shown in Fig. 5. Logically, all servers can communicate with each other by links of their shortest routes. For example, London is connected to Luxembourg via Brussels. We assume that groups of 1000 and 100 users are uniformly distributed in each square area, as shown in Fig. 5. The delay of a link between servers is considered to be proportionate with the distance of transmission and treated as the distance that is routed on the shortest path at COST239 [15] and JPN Kanto region [16]. The delay of a link between a user and a server is considered to be proportionate with the distance of transmission and treated as the distance of straight length between two-dimension coordinates.

We evaluate the delay and the computation time of each scheme depends on the maximum status holding time, $L_{\mathrm{APL}}$ and $D_i$. For COST239, we set $V_{\mathrm{U}} = 100$ and 1000, $V_{\mathrm{S}} = 11$, $E_{\mathrm{US}} = 1100$ and 11000, $E_{\mathrm{SS}} = 55$, and $Y_{\mathrm{MAX}}$=11. For JPN, we set $V_{\mathrm{U}}$= 100 and 1000, and $V_{\mathrm{S}} = 8$. We set $E_{\mathrm{US}}$=800 and 8000, $V_{\mathrm{S}}$=8, $E_{\mathrm{SS}}$=28, and $Y_{\mathrm{MAX}}$=8.

### 6.1 Evaluation with Application Maximum Status Holding Time, $L_{\mathrm{APL}}$

To evaluate delay considering application maximum status holding time, the value of $L_i$ is set to be larger than $L_{\mathrm{APL}}$. Figures 6 and 7 show the delay versus the maximum status holding time of the proposed and conventional schemes for COST239 and JPN networks, respectively.




(a) Location of servers of COST239 and users.

(b) Location of servers of JPN and users.

**Fig. 5** Location of servers and users.

For COST239, $L_i, \forall i \in V_{\mathrm{S}}$, is set to 3000 ms and $L_{\mathrm{APL}}$ is set from 1200 ms to 2200 ms. As shown in Fig. 6, the delays of the proposed scheme are reduced compared to that of the conventional scheme for 100 and 1000 users when $L_{\mathrm{APL}}$ is 1400 ms or more. The delays of the proposed scheme are less than 25 percent compared to that of the conventional scheme at the region, where $L_{\mathrm{APL}}$ is 2000 ms or more. These reductions of delay presented in Fig. 6 is achieved under the special condition that the value of $L_{\mathrm{APL}}$ becomes sufficiently large.

For JPN, $L_i$ is set to 2.0 ms and $L_{\mathrm{APL}}$ is set from 1.3 ms to 1.7 ms. As shown in Fig. 7, the delays of the proposed scheme are reduced compared to that of the conventional scheme for 100 and 1000 users when $L_{\mathrm{APL}}$ is 1.4 ms or more. The delays of the proposed scheme are less than 60

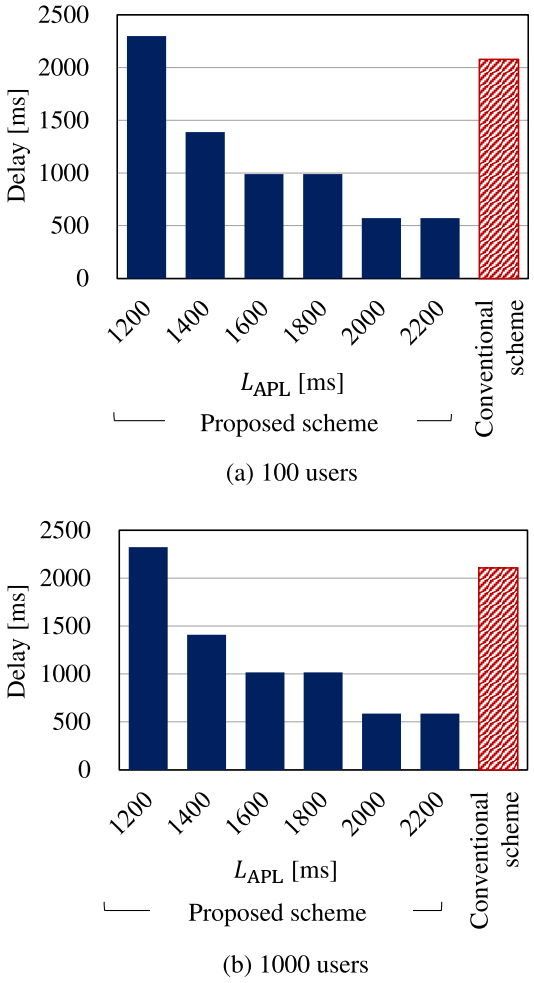



**Fig. 6** Delay for COST239 at 100 and 1000 users.

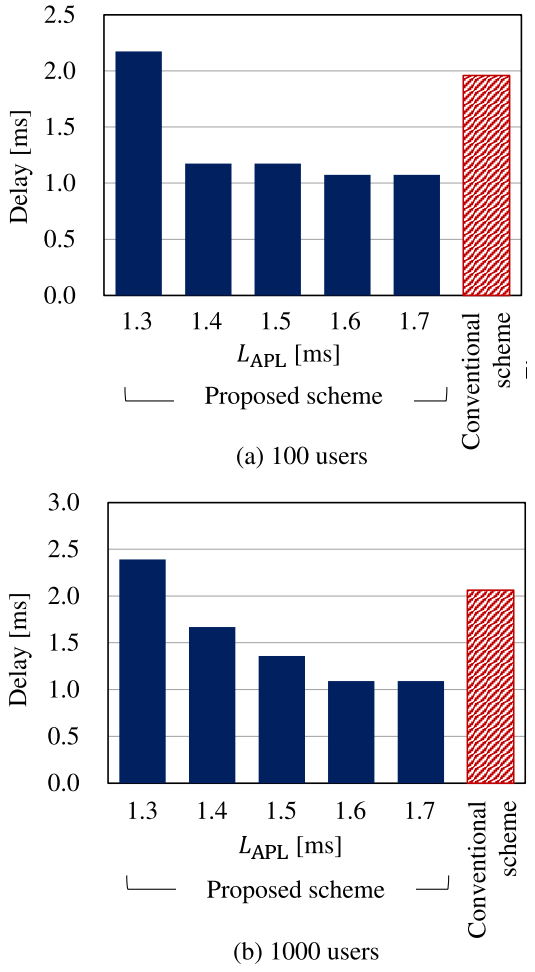



**Fig. 7** Delay for JPN at 100 and 1000 users.

percent compared to that of the conventional scheme at the region where $L_{APL}$ is 1.6 ms or more. In JPN, the difference of delay between the proposed and conventional schemes is several milliseconds. In this evaluation, the nodes are located at JPN Kanto region. The proposed scheme is expected to be effective for real-time applications if nodes are located at a wide area network, such as a nationwide area network in Japan.

As $L_{APL}$ becomes small, selecting the nearest server by each user does not satisfy the maximum status holding time constraint in (1f). In this situation, a user needs to connect to a further server so that (1f) can be satisfied; the delay between the server connected with the user and any other server connected with any other user can be reduced. As a result, $D_{US}^{max}$ becomes large. In Figs. 6 and 7, there is small difference in the delay between 100 users and 1000 users. Since the delay is determined as the maximum delay of the selected links between users and servers, the user farthest from the server determines the delay. The reason of small difference in delay between 100 users and 1000 users is that the farthest user locations of 100 users and 1000 users are near, as shown in Fig. 5. From these results, the proposed scheme is effective in reducing delay under the condition that $L_{APL}$, where users tend to select their nearest servers, is large enough. The proposed scheme is also effective for reducing delay, regardless of network topologies.

**Table 2** Average computation time for each scheme for COST239.

| No. of users | Synchronization algorithm [sec] $L_{APL}$ 1200 | 1400 | 1600 | 1800 | 2000 | 2200 | Conventional scheme [sec] |
|---|---|---|---|---|---|---|---|
| 100 | 0.8 | 1.2 | 1.6 | 0.9 | 0.2 | 0.2 | 4.4 |
| 1000 | 191.7 | 203.9 | 7.9 | 40.3 | 14.8 | 5.0 | 97.4 |

Tables 2 and 3 show the average time for five times calculation obtained by solving ILP, using both schemes for COST239 and JPN. When 100 users are considered, the computation time of the proposed scheme is smaller than that of the conventional scheme. When 1000 users are considered for COST239, the computation time of the proposed scheme is larger than that of the conventional scheme at $L_{APL}$=1200 and 1400. Considering 1000 users for JPN, the computation time of the proposed scheme is larger than that of the conventional scheme at $L_{APL}$=1.3.

The computation time considering 1000 users is within a few minutes, depending on the network topology and the

**Table 3** Average computation time for each scheme for JPN.

| No. of users | Synchronization algorithm [sec] $L_{APL}$ | | | | | Conventional scheme [sec] |
|---|---|---|---|---|---|---|
| | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 | |
| 100 | 3.4 | 1.3 | 0.6 | 0.5 | 0.6 | 2.5 |
| 1000 | 121.3 | 48.9 | 24.7 | 13.7 | 34.1 | 61.3 |

**Table 4** Values of $M_i$ and $L_i$ at defined conditions for COST239.

| Condition name | $M_i$ | $L_i$ |
|---|---|---|
| COST-A | 1000, $\forall i \in V_S$ | 1200, $\forall i \in V_S$ |
| COST-B | 1000, $\forall i \in V_S$ | 1200 ($i$=1, 3, 5, 6, 7, 8, 9, 10, 11), 1400 ($i$=2, 4) |
| COST-C | 1000, $\forall i \in V_S$ | 1200 ($i$=1, 3, 5, 7, 9, 10, 11), 1400 ($i$=2, 4, 6, 8) |
| COST-D | 153, $\forall i \in V_S$ | 1600, $\forall i \in V_S$ |
| COST-E | 153, $\forall i \in V_S$ | 1600 ($i$=1, 3, 5, 6, 7, 8), 2000 ($i$=2, 4) |
| COST-F | 153, $\forall i \in V_S$ | 1600 ($i$=1, 3, 5, 7), 2000 ($i$=2, 4, 6, 8) |

constraints of $L_{APL}$. In several cases, it is acceptable to wait a few minutes before starting a service. The locations of participating users are known in advance; the computation time of several minutes is acceptable, as the calculation can be processed before the application starts.

Figure 8 shows the delays using the proposed and conventional schemes for JPN and COST239 when 1000 users participate; the maximum number of users who select server $i$ is limited. $M_i$ at each server for COST239 and JPN is set to 153. The delay of users is obtained under the same condition as that presented in Figs. 6 and 7 without the limitation of $M_i$. The delays for COST239 at $L_{APL}$=1200 and JPN at $L_{APL}$=1.3 and 1.4 cannot be obtained, the proposed scheme does not give an answer because of constraint of $\min(L_{APL}, L_i)$. There is no server selection whose maximum status holding time is within $L_{APL}$, due to the limitation on the maximum number of users who select each server. As shown in Fig. 8, the proposed scheme is also effective for reducing delay with the limitation of $M_i$.

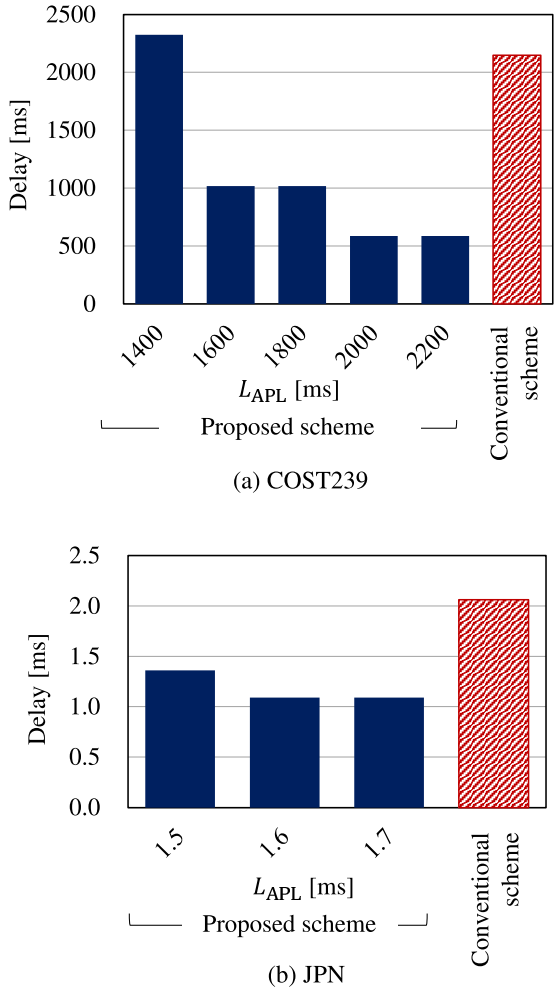



**Fig. 8** Delay for COST239 and JPN at 1000 users with $M_i$ = 153, $\forall i \in V_S$.

## 6.2 Evaluation with Server Maximum Status Holding Time, $L_i$

To evaluate delay considering the server maximum status holding time, the value of $L_{APL}$ is set to be larger than $L_i, \forall i \in V_S$. In this evaluation, we evaluate whether the delay could be reduced by increasing the number of servers with a large $L_i$ value, that is, a large amount of memory resources. Tables 4 and 5 show values of $M_i$ and $L_i$ used in this evaluation. Figures 9 and 10 show the delay of 1000 users versus the maximum status holding times of the proposed and conventional schemes for COST239 and JPN networks, respectively.

For COST239, $L_{APL}$ is set to 3000 ms and $L_i$ is set the values at Table 4. Figures 9(a) and (b) show the delay under the condition of no limitation of $M_i$ and $M_i$=153, $\forall i \in V_S$, respectively. The first bar of the left-side in Fig. 9(a) is the delay under the condition for COST-A, i.e, all servers are the same conditions. The second bar of the left-side is the delay

**Table 5** Values of $M_i$ and $L_i$ at defined conditions for JPN.

| Condition name | $M_i$ | $L_i$ |
|---|---|---|
| JPN-A | 1000, $\forall i \in V_S$ | 1.3, $\forall i \in V_S$ |
| JPN-B | 1000, $\forall i \in V_S$ | 1.3 ($i$=1, 3, 5, 6, 7, 8), 1.5 ($i$=2, 4) |
| JPN-C | 1000, $\forall i \in V_S$ | 1.3 $i$=1, 3, 5, 7), 1.5 ($i$=2, 4, 6, 8) |
| JPN-D | 153, $\forall i \in V_S$ | 1.4, $\forall i \in V_S$ |
| JPN-E | 153, $\forall i \in V_S$ | 1.4 ($i$=1, 3, 5, 6, 7, 8), 1.7 ($i$=2, 4) |
| JPN-F | 153, $\forall i \in V_S$ | 1.4 ($i$=1, 3, 5, 7), 1.7 ($i$=2, 4, 6, 8) |

under the condition for COST-B, i.e, servers 2 and 4 have more memory resources for holding the application status compared to other servers. The third bar of the left-side is the delay under the condition for COST-C, i.e, servers 2, 4, 6, 8, and 10 have more memory resources for holding the application status compared to other servers. The delay of Fig. 9(a) is reduced as the number of servers that have more memory resources increases. These results show that the proposed scheme effectively uses the memory resources of server to reduce delay. We set a value of $M_i$ to satisfy the condition that the difference in the number of users belonging to each server is less than twice and all users can select a

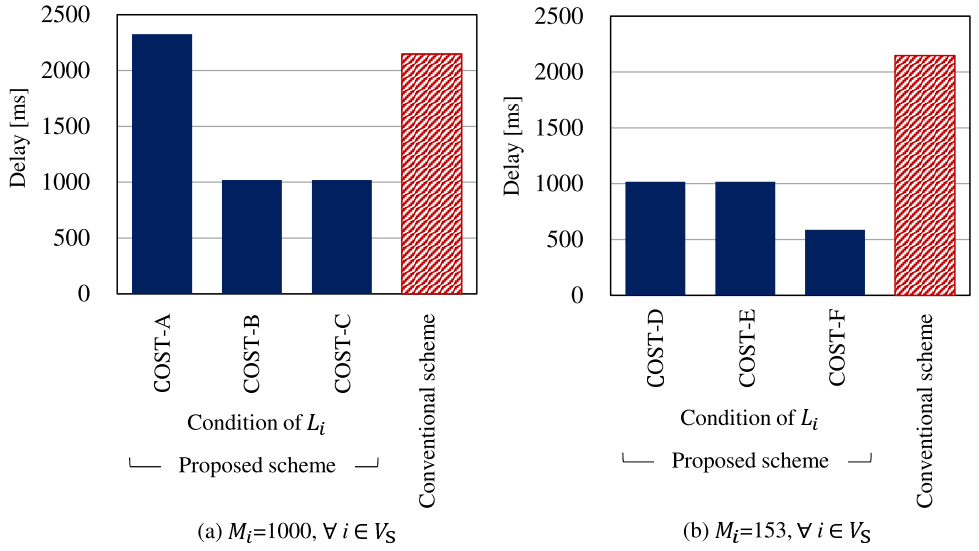



**Fig. 9** Delay for COST239 at 1000 users depending on $L_i$.

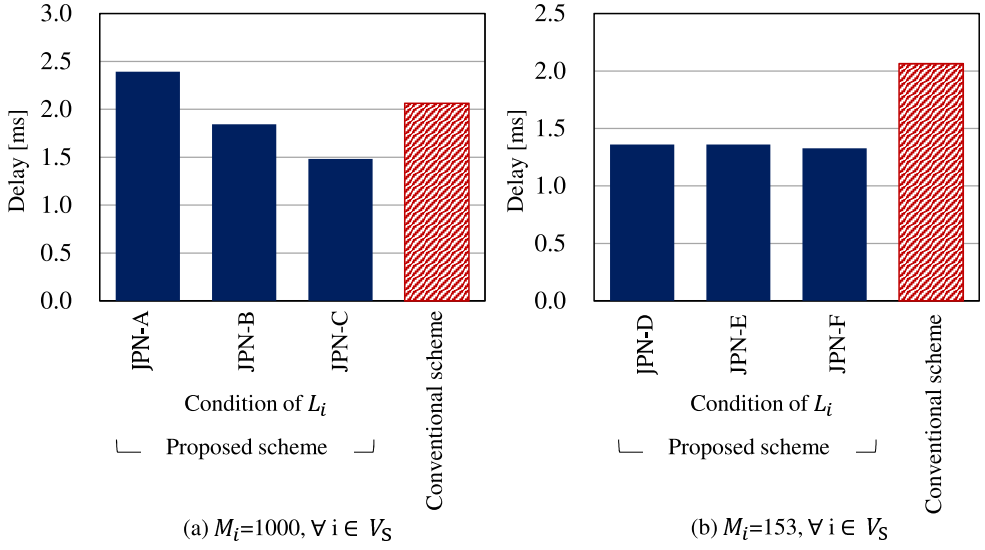



**Fig. 10** Delay for JPN at 1000 users depending on $L_i$.

server; we set $M_i = 153$.

The first bar of the left-side in Fig. 9(b) is the delay under the condition for COST-D, COST-E, and COST-F. Similar to the trend in Fig. 9(a), the delay is reduced as the number of servers that have more memory resources increases. These results show that the proposed scheme effectively uses the memory resources of server to reduce delay regardless of the limitation of $M_i$.

For JPN, $L_{\mathrm{APL}}$ is set to 3.0 ms and $L_i$ is set the values at Table 5. Figures 10(a) and (b) show the delay under the condition of no limitation of $M_i$ and $M_i$=153, $\forall i \in V_S$, respectively. Figure 10(a) is the delay under the condition for JPN-A, JPN-B, and JPN-C. The delay of Fig. 10(a) is reduced as the number of servers that have more memory resources increases. For JPN, these results also show that the proposed scheme effectively uses the memory resources of server to reduce delay, regardless of network topologies.

Figure 10(b) is the delay under the condition for JPN-D, JPN-E, and JPN-F. As shown in Fig. 10(b), the delays of the proposed scheme are not reduced as the number of servers that have more memory resources increases, but the delay of the proposed scheme is reduced compared to that of the conventional scheme. For JPN, these results show that the proposed scheme also effectively uses the memory resources of server to reduce delay regardless of the limitation of $M_i$, regardless of network topologies.

## 7. Conclusion

This paper proposed a server selection scheme based on the optimistic synchronization algorithm for a distributed processing system to minimize end-to-end delay under the con-

dition that the maximum holding time for applications and processing servers are given. The server selection problem in the proposed scheme was formulated as an ILP problem. We proved that the decision version of the server selection problem in the proposed scheme is NP-complete. Numerical results observed that the proposed scheme can reduce the delay compared to the conventional scheme, regardless of the condition that the maximum number of users who select the server are limited. These results indicated that the proposed scheme achieves a low-delay communication for network applications using an optimistic synchronization algorithm under the condition that maximum application delay and sever available memory resources are limited.

### References

[1] O. Kazuaki, K. Takeshi, and S. Katsuhiro, "A brief overview of network functions virtualisation," NTT Technical Review, vol.12, no.3, pp.1–6, 2014.
[2] ETSI TR 101 329-2:, "Telecommunications and internet protocol harmonization over networks (tiphon) release 3," End-to-end Quality of Service in TIPHON systems; Part 2: Definition of speech Quality of Service (QoS) classes, 2002.
[3] K.T. Chen, P. Huang, and C.L. Lei, "How sensitive are online gamers to network quality?," Commun. ACM, vol.49, no.11, pp.34–38, 2006.
[4] C. Bartlette, D. Headlam, M. Bocko, and G. Velikic, "Effect of network latency on interactive musical performance," Music Perception, vol.24, no.1, pp.49–62, 2006.
[5] M. Shahraeini, M.H. Javidi, and M.S. Ghazizadeh, "Comparison between communication infrastructures of centralized and decentralized wide area measurement systems," IEEE Trans. Smart Grid, vol.2, no.1, pp.206–211, 2010.
[6] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," IEEE Trans. Parallel Distrib. Syst., vol.27, no.2, pp.585–599, 2015.
[7] C. George, D. Jean, K. Tim, and B. Gordon, Distributed Systems: Concepts and Design, 5th ed., Pearson, 2011.
[8] M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," IEEE Internet Comput., vol.13, no.5, pp.10–13, 2009.
[9] R.M. Fujimoto, Parallel and Distributed Simulation Systems, Citeseer, 2000.
[10] X. Jiang, F. Safaei, and P. Boustead, "Latency and scalability: A survey of issues and techniques for supporting networked games," 2005 13th IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Communic, pp.150–155, IEEE, 2005.
[11] J.A. Miller and N.D. Griffeth, "Performance of time warp protocols for transaction management in object oriented systems," Int. J. Computer Simulation, vol.4, no.3, pp.1–34, 1994.
[12] A. Kawabata, B.C. Chatterjee, S. Ba, and E. Oki, "A real-time delay-sensitive communication approach based on distributed processing," IEEE Access, vol.5, pp.20235–20248, 2017.
[13] A. Kawabata, B.C. Chatterjee, and E. Oki, "Participating-domain segmentation based server selection scheme for real-time interactive communication," IEICE Trans. Commun., vol.E103-B, no.7, pp.736–747, July 2020.
[14] D.R. Jefferson, "Virtual time," ACM Trans. Program. Lang. Syst. (TOPLAS), vol.7, no.3, pp.404–425, 1985.
[15] M. O'Mahony, "Ultrahigh capacity optical transmission network: European research project cost 239," Information, Telecommunications, Automata Journal, vol.12, pp.33–45, 1993.
[16] "Japan photonic network model," https://www.ieice.org/cs/pn/jpn/JPNM/, 10 2020.
[17] A. Kawabata, B.C. Chatterjee, and E. Oki, "Optimal server selection scheme with optimistic synchronization for delay sensitive services," IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2021.
[18] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol.3, no.5, pp.637–646, 2016.
[19] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing," 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp.233–234, IEEE, 2015.
[20] H. Avril and C. Tropper, "Clustered time warp and logic simulation," Proc. 9th Workshop on Parallel and Distributed Simulation, pp.112–119, 1995.
[21] Y.B. Lin, B.R. Preiss, W.M. Loucks, and E.D. Lazowska, "Selecting the checkpoint interval in time warp simulation," Proc. 7th Workshop on Parallel and Distributed Simulation, pp.3–10, 1993.
[22] R.M. Karp, "Reducibility among combinatorial problems," Complexity of Computer Computations, pp.85–103, Springer, 1972.
[23] "Solving constraint integer programs," http://scip.zib.de, 10 2020.

**Akio Kawabata** is a General Manager of Network Technology Laboratories in Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan. He received the B.E., M.E. and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 1991, 1993, and 2016, respectively. In 1993, he joined Nippon Telegraph and Telephone Corporation (NTT) Communication Switching Laboratories, Tokyo, Japan, where he has been engaging to develop switching systems, and researching network design and switching system architecture. He served as a senior manager of R&D Department at NTT East from 2011 until 2014.

**Bijoy Chand Chatterjee** received the Ph.D. degree from the Department of Computer Science and Engineering, Tezpur University, in 2014. From 2014 to 2017, he was a Post-Doctoral Researcher with the Department of Communication Engineering and Informatics, The University of Electro- Communications, Tokyo, Japan, where he was involved in researching and developing high-speed flexible optical backbone networks. From 2017 to 2018, he worked as a DST Inspire Faculty at Indraprastha Institute of Information Technology Delhi (IIITD), New Delhi, India. In 2018, he joined South Asian University, New Delhi, India, where he is currently an Assistant Professor in the Department of Computer science. Before joining South Asian University, he was an ERCIM Postdoctoral Researcher at the Norwegian University of Science and Technology (NTNU), Norway. He has authored over 60 journal/conference papers. His research interests include Optical networks, QoS-aware protocols, Optimization, and Routing. Dr. Chatterjee is a Senior Member of IEEE and a Life Member of IETE.

**Eiji Oki** is a Professor at Kyoto University, Kyoto, Japan. He received the B.E. and M.E. degrees in instrumentation engineering and a Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and The University of University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn. He is a Professor at Kyoto University, Kyoto, Japan, from 2017. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks. He is an IEEE Fellow.