A Distributed Processing Communication Scheme for Real-Time Applications over Wide-Area Networks

Sanetora Hiragi
Toyohashi University of Technology
Aichi, Japan
hiragi.sanetora.jv@tut.jp

Bijoy Chand Chatterjee
South Asian University
New Delhi, India
bijoycc@ieee.org

Eiji Oki
Kyoto University
Kyoto, Japan
oki@i.kyoto-u.ac.jp

Akio Kawabata
Toyohashi University of Technology
Aichi, Japan
kawabata.akio.oc@tut.jp

***Abstract*—Low-delay networking and edge computing will enable mission-critical applications to be delivered over wide-area networks. We consider this trend to be the realization that all users can share an application space without feeling any distance difference. We propose a distributed processing scheme that keeps the order of event occurrence regardless of the distance between users and an application server. The proposed scheme can be applied to both optimistic synchronization algorithms (OSA) and conservative synchronization algorithms (CSA). In the proposed scheme, arrival events with a delay within a predefined set time (correction time) are sorted in order of occurrence before application processing. We formulate the proposed scheme as an integer linear programming (ILP) problem. The objective function of ILP consists of the number of users excluded from the delay quality, the amount of memory consumed for a rollback in OSA, and the maximum end-to-end delay. The three parts of the objective function are set weight and the sum of parts with weight is minimized. We evaluate the proposed scheme for 1000 users distributed in two types of network models. Numerical results indicate that the proposed scheme reduces memory consumption compared to that of the conventional OSA scheme. The proposed scheme works as CSA in which all events are sorted in the occurrence order if the correction time is set above the delay for the slowest event to arrive at the server.**

***Index Terms*—Delay sensitive service, distributed processing, middleware, optimistic synchronization, conservative synchronization**

I. Introduction

The launch of 5G services [1] and the advocacy of All-Photonics Network [2] by a telecommunications carrier have accelerated the realization of ultra-low-delay networks. These low-delay networks and a low-delay cloud computing infrastructure, such as edge computing [3], can provide various Internet of Things (IoT) services over wide-area networks. It is expected that these trends will accelerate and mission-critical applications will be provided. We consider this to be the realization that all users can share an application space without feeling any difference in distance from each other.

To share the space without feeling any difference in distance, an issue is a difference in delay due to the location of each user in addition to network delay. Services that do not share the space with participating users, such as video distribution, reduce network delay by caching information in the edge cloud closer to the user [4]. On the other hand, services that share the space, such as network games, may not reduce network delay by caching information in the edge cloud closer to the user due to the difference in delay for each user location. This is because the application state changes from moment to moment due to events from multiple users or the edge cloud closer to one user may be further away from other users. To overcome such delay differences for each user, shooting games take measures to judge hit decisions based on past coordinate positions [5]. Although each application takes various measures, delay differences for each user due to the location of the users for applications is a common issue for real-time IoT applications such as automatic driving and telemedicine. Therefore, the solution to this issue should work as a common function that can be used by multiple applications such as operating systems (OS) and middleware.

Research on processing events while maintaining the event order occurrence (event order guarantee) between multiple processing systems is studied in the field of parallel distributed processing. There are two typical algorithms for event order guarantee: conservative synchronization algorithm (CSA) and optimistic synchronization algorithm (OSA) [6]. In CSA, all events are sorted in occurrence order before processing the application. In OSA, the application processes events in the order of arrival. If a past event is received, the application state is rollback and modifies the processing results. CSA-based distributed processing communication scheme [7] is introduced for applications provided over wide-area networks. In this scheme, applications work on each distributed server and process all events in occurrence order. All events in each server are sorted in occurrence order and processed at the time named in virtual time. The virtual time is determined based on the maximum user-server delay. All events rearrange at a time that is delayed by the maximum user-server delay from the current time. In OSA, Time Warp [8] is a well-known scheme for the rollback process. Various types of research are also studied to reduce memory consumption which retains the application state for the rollback process. In an OSA-based distributed processing communication scheme, the network design scheme that minimizes the maximum delay is introduced at the condition of the constraint for rollback time with memory resources [9]. Thus, CSA is less burden to the application but has poor delay characteristics. OSA has good delay characteristics because events are processed in order of arrival, but it is a high burden for the application to retain the state for the rollback process.

In this paper, we propose a distributed processing communication scheme that can be commonly used for CSA and OSA applications. The proposed scheme is intended to be implemented as an OS or middleware function that can be commonly used by each application. In the proposed scheme, the correction time (T_{apl}) is introduced, and the event order guarantee is performed based on T_{apl}. The proposed scheme performs as an event order guarantee that achieves CSA when T_{apl} is set above the delay for the slowest event to arrive at the server. T_{apl} is set smaller than the delay, the proposed scheme performs to reduce the amount of memory consumed for rollback (memory consumption) in OSA. T_{apl} is set according to the service conditions, such as the acceptable delay for application quality (D_{cap}). The objective function of the proposed scheme consists of the number of users excluded from the constraints of D_{cap}, the memory consumption, and the maximum delay on the application usage (delay). The three parts of the objective function are set weight and the sum of parts with weight is minimized. The problem is formulated as an integer linear programming (ILP) problem to determine the network topology of users and servers.

We evaluate the proposed scheme under the condition that 1000 users are randomly distributed in a square area, and the application servers are located at the node positions of the Kanto area of JPN [10] (JPN-Kanto) and COST239 [11]. As an evaluation of the proposed scheme, excluded users from D_{cap}, memory consumption, and delay are compared with that of the conventional OSA [9]. We evaluate the proposed scheme at the application scenario; Augmented Reality (AR) / Virtual Reality (VR) application [12], Intelligent Transportation System (ITS) [12] and First-Person Shooter (FPS) game [13], with D_{cap}, of 1 [ms], 10 [ms], and 20 [ms], respectively. The numerical results show that the proposed scheme can reduce the memory consumption to 71.0% compared to OSA in the JPN-Kanto when D_{cap} is set to 10 [ms] and T_{apl} is set to 0.5 [ms]. In COST239, when D_{cap} is set to 20 [ms] and T_{apl} is set to 5 [ms], the proposed scheme can reduce memory consumption to 66.8% compared to OSA. A larger T_{apl} improves the reduction in memory consumption, but the excluded users may increase by the constraints of D_{cap} since delay also increases. From these results, the proposed scheme contributes to the reduction of memory consumption while maintaining the quality of service by setting T_{apl} appropriately. The proposed scheme can be also commonly used for CSA and OSA applications, depending on the value of T_{apl} for distributed processing communication scheme.

II. Proposed Scheme

A. Prerequisite of distributed processing and communication

We assume that the proposed scheme performs with multiple application servers located in a wide-area network. Figure 1 shows the communication between servers in the proposed scheme. Each user selects one server and is housed on that server. Each server receives events from contained users and sends the events to all other servers. In other words, each server receives all events from the users housed in other servers, and processes all user events.

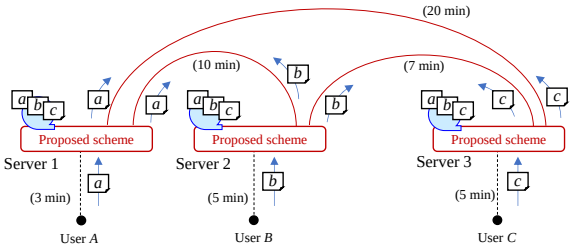


Fig. 1. Prerequisites for communication of events among servers.

B. Prerequisite of event order guarantee

In this subsection, we describe the event order guarantee for the proposed scheme. Figure 2 shows an example of the event arrival time for server 1 at the condition of the network mentioned in Fig. 1. The event arrival time in Fig. 2 is determined based on the server-to-server and user-to-server delays in Fig. 2. Events a, b, and c occur at 12:10, 12:05, and 12:00, respectively. Events a, b, and c arrive at Server 1 at 12:13, 12:20, and 12:25, respectively. The arrival order of events at Server 1 is reversed from the occurrence order according to the network delay. The proposed scheme sorts all events whose delay is within T_{apl} in occurrence order. T_{apl} is set according to the service condition, etc. Figures 3(a) and (b) explain the event order guarantee at server 1 under the delay condition in Fig. 2. Actual time + T_{apl} in the figures indicates the event notification time for application after the event order guarantee process of the proposed scheme. Figure 3(a) shows the case where T_{apl} is 25 [min]. If the delay of event n is d_n, event n waits for $T_{\mathrm{apl}} - d_n$. Event a, which has a delay of 3 [min], waits for 22 (=25-3) [min] and arrives at Server 1 at the event occurrence time + 25 [min]. Event c, which has a delay of 25 [min], waits for 0 (=25-25) [min] and arrives at Server 1 at the event occurrence time + 25 [min]. Events whose delay is within T_{apl} are sorted in the event occurrence time + T_{apl} (25 [min]) and then arrive at the application. Figure 3(b) shows the case where T_{apl} is 12 [min]. Event a, which has a delay of 3 [min], waites for 9 (=12-3) [min] and arrives at Server 1 at the event occurrence time + 12 [min]. Event c, which has a delay of 25 [ms], has a wait time of -13 (=12-25) [min]. This means that event c cannot arrive at the application within 12 [min]. Therefore, event c cannot be sorted in occurrence order and arrives 13 [min] later than T_{apl}. Events with a delay (d_n) greater than T_{apl} arrive at the application with a delay of (d_n-T_{apl}) and are rollbacked. The proposed scheme performs as the event order guarantee that realizes CSA when T_{apl} is set above the delay for the slowest event to arrive at the server.

Figure 4 shows the rollback process in the case of Fig. 3(b). When event c arrives, the processing result is corrected by the rollback process. When events a and b arrive at the server, the order of events a and b is a reversal from the occurrence order. However, the proposed scheme performs the events

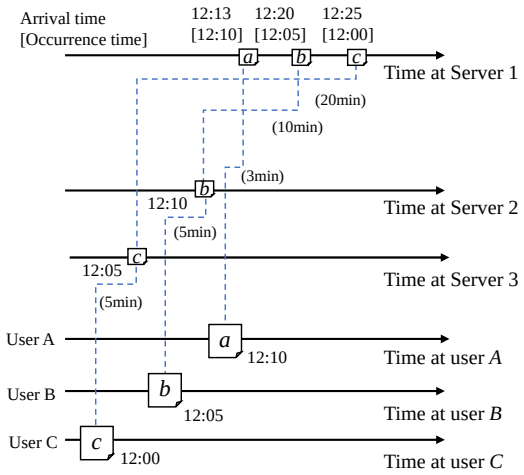
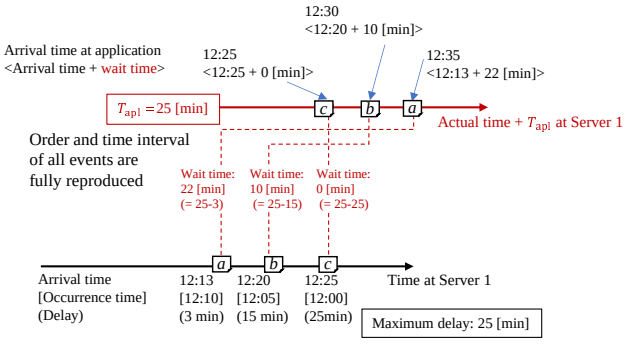
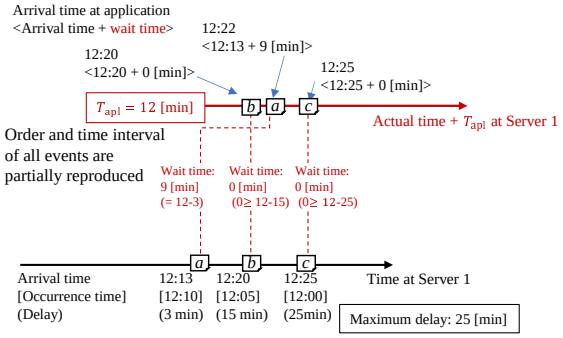


Fig. 2. Example of event arrival time at server.



(a) Event order correction in case of $T_{apl} \geq$ Maximum delay



(b) Event order correction in case of $T_{apl} <$ Maximum delay

Fig. 3. Examples of event order correction.

order guarantee before application processing, and the rollback process is not needed.

C. Delay and rollback process

This subsection describes the delay and the rollback process in the proposed scheme. $D_{\mathrm{U}}^{\max}$ is the maximum user-server delay to which the user belongs. $D_{\mathrm{S}}^{\max}$ is the maximum server-server delay. As shown in Fig. 2, an event goes through multiple servers to arrive at the destination server. The delay for the slowest event to arrive at the destination server is

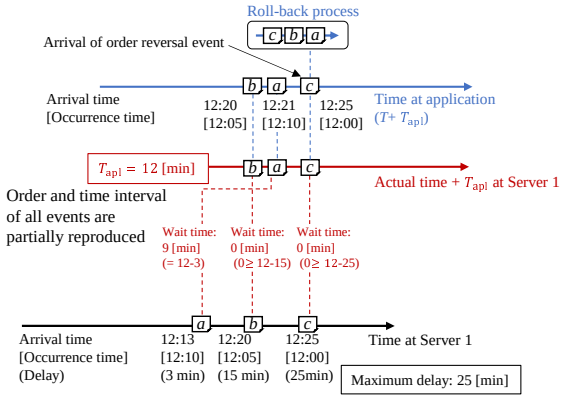


Fig. 4. Examples of rollback process in case of Fig. 3(b).

$D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$. In other words, if T_{apl} is $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$, the order guarantee for all events is possible. We describe the differences between the proposed scheme and the conservative synchronization algorithm (CSA) [7] and the optimistic synchronization algorithm (OSA) [9]. Table I shows the delay, D^{total}, and the maximum time to retain the application state for rollback (rollback time). D^{total} indicates the maximum application operation delay from the perspective of users. In the proposed scheme, D^{total} varies depending on T_{apl}. If events arrive at the server within T_{apl}, the proposed scheme guarantees the order of events. If events arrive at the server over T_{apl}, the events are processed upon arrival at the application. As shown in Table I, D^{total} is the larger value of $T_{\mathrm{apl}} + D_{\mathrm{U}}^{\max}$ or $2D_{\mathrm{U}}^{\max}$. If $D_{\mathrm{U}}^{\max}$ is within T_{apl}, the delay is $T_{\mathrm{apl}} + D_{\mathrm{U}}^{\max}$. This is because the maximum delay of arriving at the server is T_{apl} and the maximum delay of sending from the server is $D_{\mathrm{U}}^{\max}$. If the $D_{\mathrm{U}}^{\max}$ is over T_{apl}, the delay is $2D_{\mathrm{U}}^{\max}$ since the maximum delay of arriving at the server is $D_{\mathrm{U}}^{\max}$ and the maximum delay of sending from the server is $D_{\mathrm{U}}^{\max}$. As shown in Table I, the rollback time is $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} - T_{\mathrm{apl}}$, since the events whose delay is within T_{apl} are sorted in order of occurrence. In CSA, events arriving within the maximum delay, $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$, are guaranteed event order. The waiting time to send the processing results is controlled so that D^{total} is $2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$. In OSA, events are processed as soon as they arrive. Therefore, D^{total} is a round-trip delay ($2D_{\mathrm{U}}^{\max}$). The rollback time is $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$.

TABLE I
MAXIUM DELAY FOR USERS AND ROLLBACK TIME OF THE PROPOSED AND CONVENTIONAL SCHEMES.

	Delay (D^{total})	Rollback Time
Prop. ($T_{\mathrm{apl}} > D_{\mathrm{U}}^{\max}$)	$T_{\mathrm{apl}} + D_{\mathrm{U}}^{\max}$	$D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} - T_{\mathrm{apl}}$
Prop. ($T_{\mathrm{apl}} \leq D_{\mathrm{U}}^{\max}$)	$2D_{\mathrm{U}}^{\max}$	
CSA [7]	$2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$	0
OSA [9]	$2D_{\mathrm{U}}^{\max}$	$D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$

D. Formulation

In this section, we formulate the proposed scheme as an ILP problem. The acceptable delay for application quality (acceptable delay) is D_{cap}. The user selects a server so that D_{total} does not exceed D_{cap}. We set the number of excluded users as the objective function since high delay may be excluded due to the restriction of D_{cap}. In this evaluation, the amount of memory required for rollback (memory consumption) is treated as the rollback time. This is because the smaller the rollback time can reduce the memory for retaining the application state. The user selects one optimal server from multiple candidate servers. Table II shows the given parameters and the decision variables. The first objective function is the number of excluded users due to D_{cap}, which is denoted by U^{excl}. The second objective function is the memory consumption, $D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}-T_{\mathrm{apl}}$. The third objective function is the delay, D^{total}. We aim to minimize the sum of the weights of the three parts of the objective function; a weight for each part is set.

TABLE II
GIVEN PARAMETERS AND DECISION VARIABLES OF THE FORMULATED OPTIMIZATION PROBLEM.

Given Parameters	Description
T_{apl}	Correction time
D_{cap}	Acceptable delay for application quality
d_{pi}	Delay between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$
d_{ij}	Delay between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}} \setminus \{i\}$
Decision Variable	**Description**
$D_{\mathrm{U}}^{\max}$	Maximum user-server delay
$D_{\mathrm{S}}^{\max}$	Maximum sever-server delay
D^{total}	Maximum user-application delay
U^{excl}	Number of excluded users
x_{kl}	$x_{kl}=1$ if link $(k,l) \in E$ is selected, and $x_{kl}=0$ otherwise
y_i	$y_i=1$ if server $i \in V_{\mathrm{S}}$ is selected, and $y_i=0$ otherwise
u_p	$u_p=1$ if user $p \in V_{\mathrm{U}}$ is excluded, and $u_p=0$ otherwise

The network model is an undirected graph $G(V,E)$. V and E are the set of nodes and the set of undirected links, respectively. The set of users is denoted by V_{U} and $p \in V_{\mathrm{U}}$. The set of servers is denoted by V_{S} and $i \in V_{\mathrm{S}}$. The node is either user or server, so $V_{\mathrm{U}} \cup V_{\mathrm{S}} = V$ and $V_{\mathrm{U}} \cap V_{\mathrm{S}} = \emptyset$. The set of user-server links is denoted by E_{U}. $(p,i) \in E_{\mathrm{U}}$ is the link between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$. The set of sever-server links is denoted by E_{S}. $(i,j) \in E_{\mathrm{S}}$ is the link between sever $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}} \setminus \{i\}$. The link is either user-server and server-server link, so $E_{\mathrm{U}} \cup E_{\mathrm{S}} = E$ and $E_{\mathrm{U}} \cap E_{\mathrm{S}} = \emptyset$. d_{pi} is the delay of the link between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$. $D_{\mathrm{U}}^{\max}$ is the maximum delay among the selected user-server links. d_{ij} is the delay of the link between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}} \setminus \{i\}$. $D_{\mathrm{S}}^{\max}$ is the maximum delay among the selecetd server-server links. x_{kl} is a binary variable for link $(k,l) \in E$, where x_{kl}=1 if link (k,l) is selected, and x_{kl}=0 otherwise. y_i is a binary variable for server $i \in V_{\mathrm{S}}$, where y_i=1 if server i is selected, and y_i=0 otherwise. u_p is a binary variable for user $p \in V_{\mathrm{U}}$, where u_p=1 if user p is excluded and u_p=0 otherwise. The proposed scheme is formulated as follows equations.

$$\text{Objective} \quad \min\{U^{\mathrm{excl}} + \alpha(D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} - T_{\mathrm{apl}}) + \beta D^{\mathrm{total}}\} \tag{1a}$$

$$\text{s.t.} \quad \sum_{i:(p,i)\in E_{\mathrm{U}}} x_{pi} = 1, \forall p \in V_{\mathrm{U}} \tag{1b}$$

$$\sum_{p \in V_{\mathrm{U}}} u_p \le U^{\mathrm{excl}} \tag{1c}$$

$$d_{pi}(x_{pi} - u_p) \le D_{\mathrm{U}}^{\max}, \forall (p,i) \in E_{\mathrm{U}} \tag{1d}$$

$$d_{ij} x_{ij} \le D_{\mathrm{S}}^{\max}, \forall (i,j) \in E_{\mathrm{S}} \tag{1e}$$

$$D_{\mathrm{U}}^{\max} + T_{\mathrm{apl}} \le D^{\mathrm{total}} \tag{1f}$$

$$2D_{\mathrm{U}}^{\max} \le D^{\mathrm{total}} \tag{1g}$$

$$D^{\mathrm{total}} \le D_{\mathrm{cap}} \tag{1h}$$

$$y_i \ge x_{pi}, \forall p \in V_{\mathrm{U}}, i \in V_{\mathrm{S}} \tag{1i}$$

$$y_i + y_j - 1 \le x_{ij}, \forall (i,j) \in E_{\mathrm{S}} \tag{1j}$$

$$x_{ij} \le y_i, \forall i \in V_{\mathrm{S}}, (i.j) \in E_{\mathrm{S}} \tag{1k}$$

$$x_{ij} \le y_j, \forall j \in V_{\mathrm{S}}, (i,j) \in E_{\mathrm{S}} \tag{1l}$$

$$x_{kl} \in \{0,1\}, \forall (k,l) \in E_{\mathrm{S}} \cup E_{\mathrm{U}} \tag{1m}$$

$$y_i \in \{0,1\}, \forall i \in V_{\mathrm{S}} \tag{1n}$$

$$u_p \in \{0,1\}, \forall p \in V_{\mathrm{U}} \tag{1o}$$

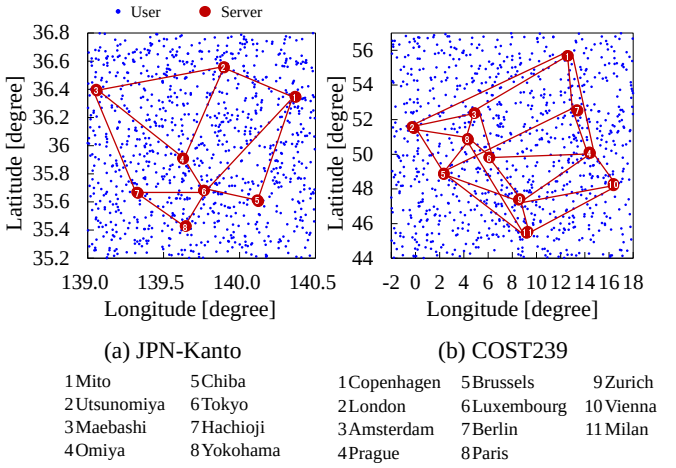
Equation (1a) indicates that the objective function with three parts, which are U^{excl}, $D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}-T_{\mathrm{apl}}$, and D^{total}. Set α and β to be $U^{\mathrm{excl}} \gg \alpha(D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}-T_{\mathrm{apl}}) \gg \beta D^{\mathrm{total}}$. Equation (1b) indicates that each user selects one server from candidate servers. Equation (1c) indicates that the number of excluded users is U^{excl}. Equation (1d) indicates that $D_{\mathrm{U}}^{\max}$ is the maximum delay for the selected link $(p,i) \in E_{\mathrm{U}}$. Equation (1e) indicates that $D_{\mathrm{S}}^{\max}$ is the maximum delay for the selected link $(i,j) \in E_{\mathrm{S}}$. Equations (1f)-(1g) indicate that D^{total} is the larger of $T_{\mathrm{apl}} + D_{\mathrm{U}}^{\max}$or $2D_{\mathrm{U}}^{\max}$. Equation (1h) indicates that D^{total} does not exceed D_{cap}. Equation (1i) indicates that if link $(p,i) \in E_{\mathrm{U}}$ is selected, server $i \in V_{\mathrm{S}}$ is selected. Equations (1j)-(1l) are a linear expression for $y_i * y_j = x_{ij}$. Equations (1m)-(1o) indicate that x_{kl}, y_i, and u_p are binary variables.

III. NUMERICAL EVALUATION AND DISCUSSION

This section investigates the performance of the proposed scheme. We evaluate the number of excluded users, memory consumption, and the delay, and compare those to the conventional OSA scheme. The parameters of α and β in Eq. (1a) are set to 0.01 and 0.00001, respectively. We use IBM ILOG CPLEX Optimization Studio (CPLEX) [14] to solve ILP problems. The computer is configured with the Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz with 28 cores. The amount of memory is 128 Gbytes. As shown in Eq. (2), the memory reduction ratio for the proposed scheme compared to

that of the conventional OSA scheme is expressed as R. We denote $D_{\mathrm{U}}^{\max}$ and $D_{\mathrm{S}}^{\max}$ in the proposed scheme as $D_{\mathrm{U}}^{\max,\mathrm{Pro}}$ and $D_{\mathrm{S}}^{\max,\mathrm{Pro}}$, receptively. We also denote $D_{\mathrm{U}}^{\max}$ and $D_{\mathrm{S}}^{\max}$ in OSA as $D_{\mathrm{U}}^{\max,\mathrm{OSA}}$ and $D_{\mathrm{S}}^{\max,\mathrm{OSA}}$, receptively.

$$R = \frac{D_{\mathrm{U}}^{\max,\mathrm{Pro}} + D_{\mathrm{S}}^{\max,\mathrm{Pro}} - T_{\mathrm{apl}}}{D_{\mathrm{U}}^{\max,\mathrm{OSA}} + D_{\mathrm{S}}^{\max,\mathrm{OSA}}} \tag{2}$$



(a) JPN-Kanto

1 Mito
2 Utsunomiya
3 Maebashi
4 Omiya
5 Chiba
6 Tokyo
7 Hachioji
8 Yokohama

(b) COST239

1 Copenhagen
2 London
3 Amsterdam
4 Prague
5 Brussels
6 Luxembourg
7 Berlin
8 Paris
9 Zurich
10 Vienna
11 Milan

Fig. 5. Location of servers and users.

We evaluate the proposed scheme with two different topologies. The candidate servers are located at nodes of the JPN-Kanto [10] and nodes of COST239 [11], as shown in Fig. 5. There are 1000 users in each topology. In JPN-Kanto, there are eight candidate servers, and 1000 users are randomly distributed in an area with a latitude of 139 degrees to 140.4 degrees and a longitude of 35.2 degrees to 36.8 degrees. In COST239, there are 11 candidate servers, and 1000 users are randomly distributed in an area with a latitude of -2 degrees to 18 degrees and a longitude of 44 degrees to 57 degrees. Network delay usually includes transmission delay, processing delay in switches, and queuing delay during congestion. In this evaluation, the netowork delay is treated as transmission delay since it tends to be proportional to transmission distance. The delay is assumed to include the processing delay and the queuing delay. The transmission delay per 1000 [km] is assumed to be 5 [ms]. The user-server delay is proportional to the linear distance in coordinates and three times the linear distance. This is because the metro network often is composed of ring topologies in the actual network. We assume that the proposed scheme is provided at quality-guaranteed networks for delay-sensitive services. The network guarantees the delay based on service level agreement (SLA) [15]. We configure D_{cap} for three types of AR/VR [12], ITS [12], and FPS games [13]. D_{cap} in the AR/VR, ITS, and FPS are set for 1 [ms], 10 [ms], and 20 [ms], respectively [12]- [13].

We evaluate the number of excluded users, U^{excl}, and the memory reduction rate, R, of the proposed scheme and compare it to the that of the conventional OSA scheme. We set T_{apl} in the range from 0 [ms] to D_{cap} [ms]. Figure 6 shows the results for JPN-Kanto with D_{cap} set to 1 [ms] and 10 [ms]. Figure 7 shows the results for COST239 with D_{cap} set to 10 [ms] and 20 [ms]. In Figs. 6 and 7, the first vertical axis is R, the second vertical axis is U^{excl}, and the horizontal axis is T_{apl}. As shown in Figs. 6 and 7, R tends to decrease as T_{apl} increases, since the memory consumption (=rollback time), $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} - T_{\mathrm{apl}}$, decreasing by increasing T_{apl}. In Figs. 6(a), (b), 7(a), and (b), R is 0 at T_{apl} = 1 [ms], 2 [ms], 10 [ms], and 15 [ms], respectively. These results indicate that the rollback time, $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} - T_{\mathrm{apl}}$, is reduced by increasing the correction time, T_{apl}. In other words, if T_{apl} is set above the rollback time, $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$, the rollback process is unnecessary. In Figs. 6(a), (b), 7(a), and (b), R is 100% at T_{apl} = 0 [ms]. These results indicate that the proposed scheme works as OSA when T_{apl} = 0 [ms].

As shown in Figs. 6 and 7, U^{excl} increases by increasing T_{apl}, since the delay increases by T_{apl} increasing, and the users with a high user-server delay are excluded to satisfy the delay constraint. As shown in Figs. 6(a) and (b), if T_{apl} and the location of users are the same, U^{excl} increases when D_{cap} is small. As shown in Figs. 6(b) and 7(a), if D_{cap} and T_{apl} are the same, U^{excl} increases when users are widely distributed (high delay). From these results, the proposed scheme contributes to reducing the memory consumption in OSA and behaves in CSA when T_{apl} is set above $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$.

We discuss the proposed scheme in terms of R and D^{total}. Figure 8(a) shows R and D^{total} of the proposed scheme under the same conditions as Fig. 6. When D_{cap} is 10 [ms] in Fig. 8(a), R is 71.0% at T_{apl} = 0.5 [ms]. When D_{cap} is 1 [ms] in Fig. 8(a), U^{excl} is 244, 244, and 1000 at T_{apl} = 0.15 [ms], 0.5 [ms], and 5 [ms], respectively. This is because a larger T_{apl} may increase the excluded users because of the larger delay. When D_{cap} are 1 [ms] and 10 [ms] in Fig. 8(a), R decreases with increasing T_{apl} because the rollback time is reduced by the increase in T_{apl}. D^{total} increases with increaseing T_{apl} if T_{apl} is greater than $D_{\mathrm{U}}^{\max}$. D^{total} is equal to $2D_{\mathrm{U}}^{\max}$ if T_{apl} is smaller than $D_{\mathrm{U}}^{\max}$. A smaller D_{cap} becomes stringent for the delay constraint and the users with high delay may be excluded. From these results, the proposed scheme requires T_{apl} to be set in consideration of D_{cap}.

Figuer 8(b) shows R and D^{total} of the proposed scheme under the same conditions as Fig. 7. When D_{cap} is 20 [ms] in Fig. 8(b), R is 66.8% at T_{apl} = 5 [ms]. When D_{cap} is 10 [ms] and T_{apl} is 5 [ms], U^{excl} is 0 and 168 in Fig. 8 (a) and Fig. 8(b), respectively. This is because COST239 has a wider distribution of users (higher delay) than JPN-Kanto and users far from the server are excluded in order to satisfy the delay constraint. The optimal T_{apl} varies with the distribution of users even if the same D_{cap}. When users are widely distributed, more users might be excluded from the delay constraints. As shown in Fig. 8(a), R decrease with increasing T_{apl}. The proposed scheme reduces memory consumption, R, compared to that of the conventional OSA scheme in two types of networks. In the proposed scheme, the optimal T_{apl} should be set carefully for the constraint of the acceptable delay, D_{cap}. The proposed scheme is effective in reducing memory consumption or excluded users compared to

that of the conventional OSA scheme due to the optimal T_{apl} considering D_{cap} and user distribution. The proposed scheme works as CSA in which all events are sorted in the occurrence order when T_{apl} is set above $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$. These results indicate that the proposed scheme can be commonly used for both CSA and OSA applications, and contributes to reducing memory consumption and expanding the number of users.

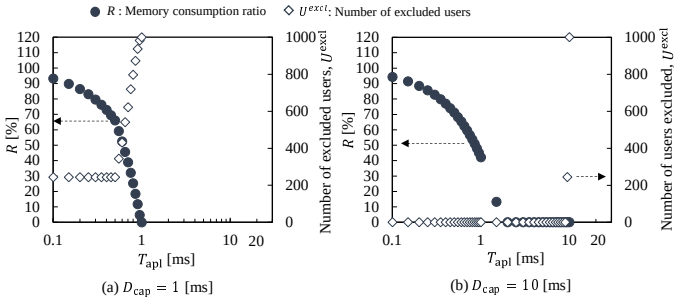


Fig. 6. Memory consumption ratio, R, and number of excluded users, U^{excl}, in JPN-Kanto.

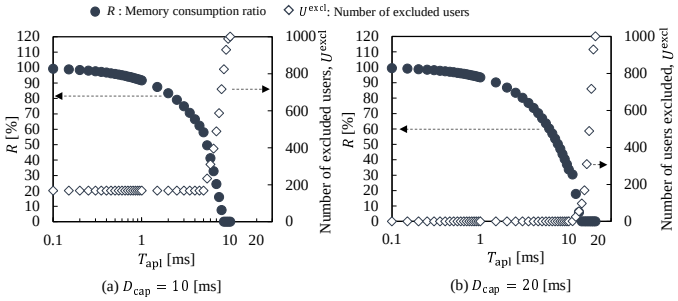


Fig. 7. Memory consumption ratio, R, and number of excluded users, U^{excl}, in COST239.

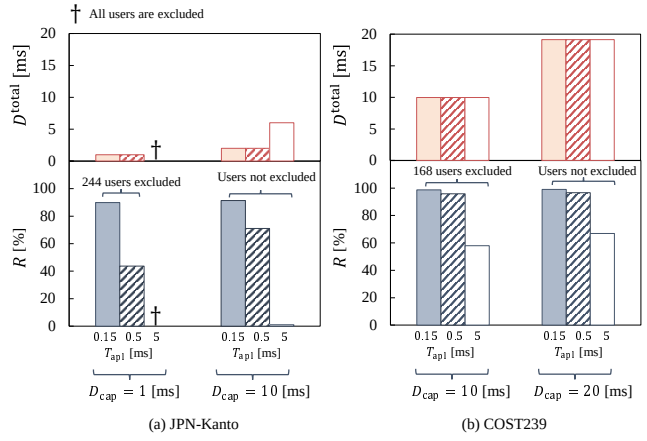


Fig. 8. Memory consumption ratio, R, and delay, D^{total}, in JPN-Kanto and COST239 of the proposed scheme.

IV. Conclusion

We proposed a distributed processing communication scheme that can be applied to both optimistic synchronization algorithms (OSA) and conservative synchronization algorithms (CSA). The proposed scheme performs event guarantee before processing the application based on T_{apl}. We evaluated the proposed scheme for 1000 users distributed in Kanto model of JPN (JPN-Kanto) and the COST239 model. From the results of the numerical evaluation, the proposed scheme reduces the memory consumption to 71.0% in JPN-Kanto, and to 66.8% in COST239 compared to that of the conventional OSA. In the proposed scheme, the optimal T_{apl} should be set carefully for the constraint of the acceptable delay and user distribution because the delay may be increased. The proposed scheme works as CSA in which all events are sorted in the occurrence order if T_{apl} is set above the delay for the slowest event to arrive at the server.

Acknowledgements

This work was supported by JSPS KAKENHI JP23K16867. We would like to thank Ryosuke Harada for his contribution to this research. The authors would like to thank Information Media Center (IMC) at Toyohashi University of Technology for providing computer resources and the kind support of its staff.

References

[1] "NTT DOCOMO to Launch 5G Service in Japan on March 25," https://www.nttdocomo.co.jp/english/info/media_center/pr/2020/0318_00.html, online; accessed 18 May 2023.

[2] A. Kawabata and Y. Aoyagi, "Network-service technology enabled by the All-Photonics Network," *NTT Technical Review*, vol. 19, no. 10, pp. 18–24, 2021.

[3] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[4] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated Web browsing," *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pp, 233–234, Mar. 2015.

[5] X. Jiang, F. Safaei, and P. Boustead, "Latency and scalability: a survey of issues and techniques for supporting networked games," *IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Commun*, vol. 1, pp. 6, 2005.

[6] R. M. Fujimoto, *Parallel and Distributed Simulation System.* New York, NY, USA: Wiley, 2000.

[7] 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.

[8] D. R. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 7, no. 3, pp. 404–425, 1985.

[9] A. Kawabata, B. C. Chatterjee, S. Ba, and E. Oki, "An optimistic synchronization based optimal server selection scheme for delay sensitive communication services," *IEICE Trans. on Commun.*, vol. 104, no. 10, pp 1277–1287, 2021.

[10] "Japan Photonic Network Model," http://www.ieice.org/cs/pn/jpn/jpnm.html, online; accessed 18 May 2023.

[11] M. O'mahony, "Ultrahigh capacity optical transmission network: European research project cost 239," *Information, Telecommunications, Automata Journal*, vol. 12, pp. 33–45, 1993.

[12] D. Feng, L. Lai, J. Luo, Y. Zhong, C. Zheng, and K. Ying, "Ultra-reliable and low-latency communications: applications, opportunities and challenges," *Science China Information Sciences*, vol. 64, pp. 1– 12, 2021.

[13] S. Host, W. Tarneberg, P. Odling, M. Kihl, M. Savi, and M. Tornatore, "Network requirements for latency-critical services in a full cloud deployment," *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, pp. 1-5, 2016.

[14] "CPLEX Optimization Studio," https://www.ibm.com/products/ilog-cplex-optimization-studio?mhsrc=ibmsearch_a&mhq=cplex, online; accessed 18 May 2023.

[15] "Global IP Network," https://www.ntt.com/en/services/network/gin/sla.html, online; accessed 18 May 2023.