Distributed Processing Network Design Scheme for Virtual Application Processing Platform

Akio Kawabata, *Member, IEEE,* Sanetora Hiragi, Bijoy Chand Chatterjee, *Senior Member, IEEE,* and Eiji Oki, *Fellow, IEEE*

***Abstract*—Delay-sensitive applications have been provided through a low-delay network utilizing multiple edge clouds. For applications that involve sharing status among multiple users, it is crucial to prevent longer communication delays for users who are farther from the application server compared to those who are closer. To address this issue, this paper proposes a distributed processing network design scheme for virtual processing platforms using low-delay networks and widely distributed servers. The proposed scheme introduces T_{apl} as a given parameter for correcting events in occurrence order. Events within T_{apl} delay are sorted in occurrence order. The proposed scheme can change its operation mode from a conservative synchronization to an optimistic synchronization depending on the setting of T_{apl}. The proposed scheme is formulated as a mixed-integer linear programming problem to determine users' and servers' distributed processing network configuration. We evaluate the proposed scheme on two different network topologies. Numerical results indicate that, depending on the setting of T_{apl}, the proposed scheme can reduce the maximum amount of memory used for rollback processes in optimistic synchronization-based applications or realize a conservative synchronization algorithm. The computation time under the condition of 1000 users is within a maximum of nine [sec], an acceptable amount of time for preparation before starting a planned service. These results indicate that the proposed scheme realizes event order correction with excellent delay characteristics and applies to virtual processing platforms.**

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

I. Introduction

INTERNET of Things (IoT) services are increasingly being delivered to wide-area users by connecting multiple clouds over a low-delay network and providing fifth-generation mobile services [1] and edge cloud computing infrastructure [2], [3]. Telecommunications carriers have announced a lower-delay network, such as all-photonics network [4]. These will deliver services such as telemedicine and automated driving, which were previously difficult to provide over the network. Among these real-time network services are applications in which multiple users share the same application state (space) and communicate bidirectionally in real-time (space-sharing applications), such as virtual reality and online trading.

In these applications, it is essential to ensure that unfairness due to differences in communication delays between users does not occur. For example, in online trading, users located far from the trading center must not be disadvantaged by users closer to the center who make a transaction more quickly. We refer to the process of mitigating such disadvantages caused by communication delay differences as delay fairness. In space-sharing applications using low-delay networks and multiple clouds, ensuring delay fairness is crucial to provide fair treatment for all users.

First, we explain the delay discussed in this paper. When users use an application, there are delays in communication and processing. In this paper, a delay between when a user sends an event and when it arrives at the application server is called a communication delay, and a delay between when the server receives the event and when the application processes the event is called a processing delay. The processing delay consists of a queuing delay from when the event arrives at the server until it is notified to the application and an application processing delay from when the application receives the event and processes it. Our research target is a network design scheme of a distributed application processing system. So, we focus on the sum of the communication delay and the queuing delay from when a user sends an event until it is notified to the application as the delay. The communication delays consist of transmission delays on optical fibers and delays due to communication systems in the network, such as routers and transmission systems, and we treat the communication delay that increases as the communication distance increases. The queuing delay is usually minimal, with sufficient processing performance on the server. We control the queuing delay to achieve fairness in delays in our research.

For the delay fairness, events from widely distributed users must be processed in the order in which they occur. It is necessary to process events so that user events with large communication delays that occur first are not overtaken by events with small communication delays that occur later. This issue of delay fairness has been addressed in the research field of parallel and distributed processing as a study of processing events in the order in which events occur across multiple processes. In this field of study, the methods for achieving delay fairness are divided into two approaches. In a conservative synchronization algorithm, events are reordered in occurrence order before application processing (event order correction). In an optimistic synchronization algorithm, events are processed in the order of arrival. If events that have occurred in the past arrive, the processing results are modified as arriving in the order of occurrence (rollback) [5].

The researchers have studied the characteristics of the two

A. Kawabata and S. Hiragi are with Toyohashi University of Technology, Aichi, Japan.
B. C. Chatterjee is with South Asian University, New Delhi, India.
E. Oki is with Kyoto University, Kyoto, Japan.
Manuscript received April 16, 2025; revised April 16, 2025.

synchronization algorithms. As the conservative synchronization algorithm performs event order correction in advance, applications can proceed with event processing in order, but the queuing delay occurs for the event order correction. From the application's perspective, this queuing delay is treated as equivalent to a communication delay. The optimistic synchronization algorithm has good delay characteristics because it processes events in the order they arrive, but memorizing the application state for rollback is necessary [6]. For example, a shooting game provided via a network incorporates processing to determine a bullet's hit by rewinding the target's coordinates to a previous time considering the communication delays [7]. A drawback of the optimistic synchronization algorithm is its high memory consumption, which is needed to store the application states for potential rollbacks.

The conservative and optimistic synchronization algorithms have distinctly different processes for the delay fairness. Each application selects each method to ensure the delay fairness according to the characteristics of the application. In the optimistic synchronization algorithm, the states to be kept for rollback vary for application. Like these, adopting different solutions for each application is inefficient in the era of space-sharing applications. If a delay fairness mechanism that can be universally applied to various applications is realized, it is expected to accelerate the deployment of space-sharing applications. This raises the following question: *can we develop a delay fairness solution that can be commonly applied to applications with different synchronization algorithms?*

Our research target is a solution to realize delay fairness that can be commonly used for various applications. Until now, a solution for synchronization algorithms has corresponded to an individual application. This paper proposes a distributed processing network design scheme for a virtual application processing platform that achieves delay fairness using a low-delay network and servers in edge clouds. The proposed scheme provides the optimal network design for multiple users and candidate servers, adapting to the application's delay requirement. The network design determines an optimal set of servers accommodating users and the connections between the users and servers.

The proposed scheme is intended to be implemented as a function of an application processing platform, such as an operating system (OS) or middleware. It is a queuing function for exchanging information between a server's network interface and an application. The proposed scheme introduces T_{apl} as a time parameter related to delay fairness. T_{apl} is the maximum queuing time for event order correction. User events arriving with the communication delay within T_{apl} are sorted in order of occurrence. Meanwhile, user events arriving with the communication delay beyond T_{apl} are not sorted in order of occurrence and must be modified to guarantee delay fairness. By varying the value of T_{apl}, it is possible to switch the mode of operation to conservative or optimistic synchronization. In the optimistic synchronization algorithm mode, it works as a function to reduce the maximum retention time for rollback by the value of T_{apl}, thus reducing memory consumption, the disadvantage of the optimistic synchronization algorithm. By setting the value of T_{apl} to the optimal value for the application, the proposed scheme can be used as the function of event order correction for various applications. This achievement will accelerate a new operating environment for space-sharing applications.

The proposed scheme is evaluated on two network topologies, COST [8] and NSFnet [9]. We evaluate the working mode transition of the proposed scheme, the number of excluded users that cannot satisfy the acceptable delay of application, and the amount of memory consumed for rollback, depending on the value of T_{apl}. In addition, we consider the constraints of the number of available servers and the maximum number of users each server can accommodate. We also evaluate the computation time to determine the network configuration in the proposed scheme. Through numerical evaluation, we show that the proposed scheme can reduce the memory consumption for rollback compared to the conventional optimistic synchronization algorithm [10], regardless of the network topology. We also indicate that distributed processing on multiple servers has better delay characteristics than centralized processing on a single server. These effects are maintained regardless of network topology and constraints of server capacity. The computation time to determine the network of users and servers is the acceptable design time before launching the service.

This paper significantly expands on the work presented in [11]. The enhancements include: (i) detailed explanations of the assumptions and use cases, (ii) a comprehensive survey of related research on conservative and optimistic synchronization algorithms, delay fairness control, and application allocation for distributed computing, (iii) an improved formulation for incorporating the parameter of the number of available servers, (iv) an enhanced method for calculating communication delay to achieve better performance, and (v) evaluations and discussions considering the number of available servers, the maximum number of users accommodated, and computation time.

The remainder of this paper is organized as follows: Section II presents related research and the position of the proposed scheme. Section III describes the proposed scheme in detail. Section IV evaluates the performance of the proposed scheme from various perspectives. Finally, Section V provides a summary.

II. Related work

This section outlines related works on delay fairness for distributed processing, followed by recent advancements in the rollback process and event order correction. Additionally, it covers application placement for multiple clouds and delay fairness in network applications. Finally, it positions the proposed scheme within the context of existing literature.

A. Dealy fairness for distributed processing

Research on processing events in occurrence order across multiple processes has been addressed in parallel and distributed processing research. In these researches, methods for achieving delay fairness can be broadly divided into conservative and optimistic synchronization algorithms [5], [10], [12].

The conservative synchronization algorithm, which performs event order correction in advance, allows event processing in the order of arrival at the application, but it incurs queuing delays for event order correction before application processing. The optimistic synchronization algorithm has good delay characteristics because events are processed in the order of arrival at the server, but requires the retention of application states for the rollback process. Time warp [6] is a solution to realize the rollback process, and the memory consumption due to state retention is an issue [13]. The conservative and optimistic synchronization algorithms have different processing methods, where one or the other is used for each application, depending on application characteristics and quality conditions.

Research has been conducted to integrate conservative and optimistic synchronization algorithms. The works [14], [15] introduced an approach to switch between conservative and optimistic synchronization depending on the situation in each event. The usual operating mode is optimistic synchronization, which rolls back and corrects processing results when past events arrive. However, if the local system's virtual time of application processing is closer than the timestamp of the most delayed event, it switches to conservative synchronization. The synchronization algorithm can be switched because all arriving events can be sorted in order of occurrence until the virtual time of application processing. This approach is generic and has a wide range of applications. In our approach, the maximum delay time of events and virtual running time of an application is fixedly determined, and conservative and optimistic synchronization is selected and fixed per application. Our research distinguishes between applications that support optimistic synchronization with a rollback mechanism and applications that use conservative synchronization with no rollback mechanism. It also targets the realization of a platform that both applications can use.

B. Recent developments in rollback process and event order correction

We present recent developments in the rollback process for the optimistic synchronization algorithm. In particular, good game peace out (GGPO) [16], a software development kit (SDK) for rollback processing in peer-to-peer games, was open-sourced in 2019. In GGPO, the application proceeds with events in order of arrival. If a previously occurring event with a large delay (remote user's event) has not arrived, the application will predict the remote player's event based on past events and proceed without waiting (speculative execution). When a previously occurring event arrives from a remote player, the event is compared to the predicted event, and if they differ, the results are corrected by the rollback process. GGPO is provided as an SDK and designed to be easily integrated into new and existing games. For the rollback process [17] introduced an efficient speculative execution method to apply to various applications to reduce the memory consumption and processing load for state retention [17]. These works for application implementation are compatible with our research on controlling the rollback time in middleware or an OS.

The related researches on event order correction are explored. Financial transactions represent a prime example of applications that demand strict event order guarantees. In delivery based ordering (DBO) [18], a method of event order guarantee was introduced in the financial trading system and was evaluated for performance. To ensure that transactions are not disadvantaged by communication delays, events are time-stamped and processed in the order of actual event occurrence by reordering events based on the time stamp before the application processes them. This work is similar to ours in that events are sorted in order of occurrence based on timestamps. Still, it does not provide a network design method to cancel the difference in each user's communication delay. Our work deals with network design between multiple processing systems, such as edge clouds, located over a wide area.

C. Application placement for multiple clouds

We discuss related research on application placement for multiple clouds (servers) in a wide-area network. In [19], various resource placement methods in virtualized networks were comprehensively investigated based on optimization parameters such as latency, server processing load, network bandwidth, and the number of service chains. In [20], a design method was presented for the optimization parameters of delay and reliability, introducing a parameter that controls the trade-off between the two optimization parameters. The work [21] addressed delay-aware resource allocation in cases where application providers can offload applications to servers in multiple locations, and users can also switch application providers. The difference between these works and our research is that we deal with delay fairness. In this aspect, our research is specific to applications that require delay fairness in which multiple users share the application state by bidirectional communication.

D. Dealy fairness in network applications

Finally, we discuss related works for event order guarantee in network applications. The work [12] introduced a conservative synchronization-based distributed processing approach that achieves event order guarantee. The work [10] effectively reduced memory consumption for rollback based on the optimistic synchronization algorithm but could not be applied to both synchronization algorithms. From these perspectives, our research provides a new value that can be used for two different synchronization algorithms.

E. Positioning of proposed scheme

Considering the other works [10], [12], [14]–[21] mentioned in the above subsections, the scope of the proposed scheme is as follows. The proposed scheme targets network applications requiring delay fairness. In such scenarios, the proposed scheme's innovative aspect lies in its ability to offer a delay fairness mechanism applicable to both optimistic and conservative synchronization through parameter settings.

III. PROPOSED SCHEME

A. Overview

The proposed scheme is a distributed processing network design scheme for space-sharing applications. The scheme is

intended to be used in a virtual application processing platform with multiple application servers distributed over a wide-area network. The scheme is used for a event queuing between a server's network interface card (NIC) and an application to support the delay fairness. $T_{\rm apl}$ is introduced as a given parameter related to delay fairness and is the maximum queuing time for event order correction. Events within $T_{\rm apl}$ delay are sorted in order of occurrence. $T_{\rm apl}$ is set to one value for each application. If the delay between user p and server i is d_{pi}, the events of user p are queued for $T_{\rm apl}$-d_{pi} before being notified to the application. Events with a delay of $T_{\rm apl}$ or less are sorted in the occurrence order, so if an application needs to process events in the occurrence order, $T_{\rm apl}$ is set as the maximum delay of the users and servers. If the maximum delay between a user and a server is 10 [ms], $T_{\rm apl}$ will also be set to 10 [ms]. Details on event sorting are described in Section III-C2. Therefore, setting $T_{\rm apl}$ to the maximum delay between users and servers, the proposed scheme works as a conservative synchronization. Setting $T_{\rm apl}$ to smaller than the maximum delay, the proposed scheme works as a function to reduce the maximum rollback time in an optimistic synchronization algorithm.

B. Assumptions

The proposed scheme operates under the following assumptions. Initially, we address the network configuration assumptions, followed by those regarding communication delay. Regarding the connection between users and servers, users can select multiple available servers from which they can select the optimal one. Telecommunication carriers' networks, such as Internet protocol (IP) networks, allow subscribed users to connect to any point in the network. We assume that users can connect to the location of any server via the carrier's network.

The communication delay is measured by a time-stamp in the received packet. The sender inserts the time information generated by network time protocol (NTP) [22] or precision time protocol (PTP) [23] as a time-stamp in the sent packet. The communication delays shall be measured in advance and re-measured periodically. As the communication delay is measured as the difference between the time of event transmission and time of reception, it includes the processing delay of network devices, such as routers or transmission systems, and the transmission delay over optical fibers. To provide real-time services, a congestion-free network with a guaranteed service level agreement (SLA) [24] is assumed. The highest measured delay is used if the delay varies depending on traffic conditions. The proposed scheme cannot be applied in networks where communication delays may become unlimited to provide real-time system services. If the communication delay temporarily increases due to a network equipment failure, etc., and the maximum delay determined by the proposed scheme needs to be changed, the application must be stopped, and the network is reconfigured based on the measurement delay.

Let us delve into server processing delay. Typically, when an application operates on servers within a network to deliver a service, the delay perceived by the user encompasses both the processing delay at the server and the communication delay. The proposed scheme deals with delays for the fairness of the application. That is, it reduces the time disadvantage caused by the user's location and network configuration. From this perspective, the proposed scheme excludes processing delays at the server and treats only communication delays. In addition, if a telecommunication carrier or service provider provides real-time system services, the equipment is constructed to minimize processing delays so that the quality of service and user operability are comfortable.

We explain the constraints to be considered for the proposed scheme. The proposed scheme is assumed to apply to applications with multiple clouds connected via a low-delay network to provide IoT services. Considering the quality conditions and operability of the application, an acceptable delay is introduced as a constraint. Users whose delays do not satisfy the constraint are excluded from service provision as excluded users. Considering the processing performance of each server, the maximum number of users connected to the server is introduced as a constraint. Considering the applications that cannot support distributed processing and the limited number of servers available in the entire network, the maximum number of available servers is introduced as a constraint. The proposed scheme determines the user-server and server-server network design under the condition that all constraints are satisfied.

C. Model

1) Communication model: In the proposed scheme, event communication with each user is processed in distributed servers. Figure 1 shows an example of event communication between users and distributed servers. The distributed processing servers multicast events received from users to other distributed processing servers. Each server receives events directly from the users it accommodates, and events from users accommodated on other servers are received via other servers. Each server receives events for all users, and the same application runs with the same events in parallel at distributed servers. The event order correction described in Section III-C2 is performed on each server. Each server produces the same result, and the application returns the same result to all users. At each distributed server, the result is returned to the users who are accommodated on their own server, and the result of the users who are accommodated on other servers is discarded.

In the example in Fig. 1, the maximum delay for an event to arrive at each of the three servers is 11 [sec] of the delay for User A's event to arrive at server 3. Because of the queuing process upon event reception described in Section III-C2, the delay until the application receives an event sent by the user is the greater value of $T_{\rm apl}$ and 11 [sec]. In addition, when the server sends the processing results to the user, a queuing process ensures that all users receive them simultaneously. The delay between the user and accommodating server is adjusted to the maximum delay between the user and server, which is 3 [sec] by the queuing process. The maximum delay when $T_{\rm apl}$=0 is shown on the left of Fig. 1.

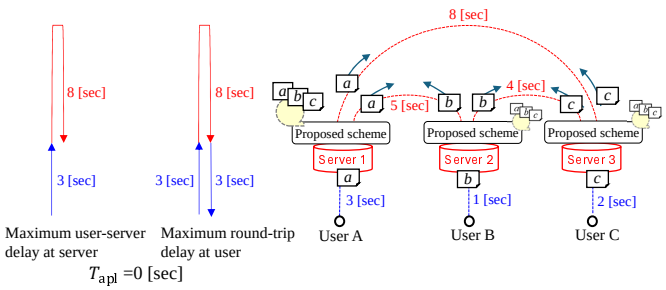


Fig. 1. Example of communication in distributed processing.

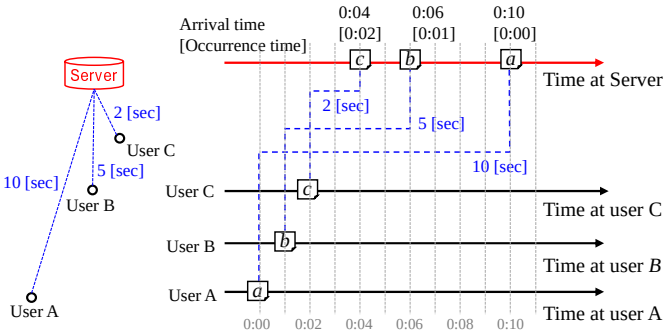


Fig. 2. Example of event reversal among users.

2) Event order correction model: We present the event order correction in the proposed scheme. The event order correction is achieved through event queuing between the server's NIC and the application. We utilize the example of processing on a single server as a straightforward model for comprehension. Even within distributed processing systems, the handling of events between users and servers remains consistent with that of a single server. Figure 2 shows an example of an event order reversal caused by a difference in communication delay. Users A, B, and C accommodate the server with communication delays of 10, 5, and 2 [sec], respectively. In the illustration on the right, time proceeds from left to right. User A, B, and C's events occur at 0:00, 0:01, and 0:02 in the order of events a, b, and c, respectively. As shown in Fig. 2, events a, b, and c arrive at the server at 0:10, 0:06, and 0:04, respectively, indicating a reversal of order due to differences in communication delay.

The order of events can be corrected in occurrence order as each event is inserted with a time-stamp of when it was sent. The proposed scheme introduces $T_{\rm apl}$ as a given parameter. $T_{\rm apl}$ is the maximum queuing time for event order correction, and the running time of the application is $T + T_{\rm apl}$, i.e., it is lagged by $T_{\rm apl}$ from the current time T.

An example of the proposed scheme working as a conservative synchronization algorithm is illustrated in Fig. 3 set with $T_{\rm apl}$=10 [sec]. In the proposed scheme, each event is passed to the application by waiting for a time equal to $T_{\rm apl}$ minus the communication delay of each user. As the communication delay before the event arrives at the server is 10, 5, and 2 [sec] for User A, B, and C, respectively, events a, b, and c

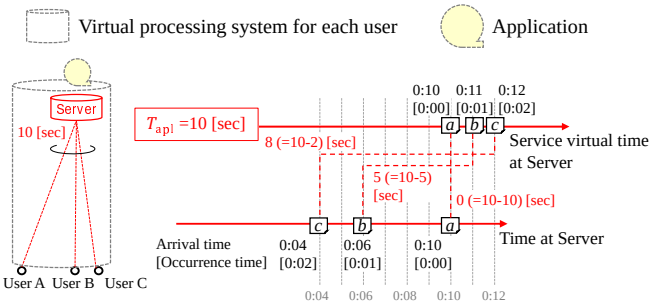


Fig. 3. Example of conservative synchronization mode in proposed scheme.

wait 0 (=10-10), 5 (=10-5), and 8 (=10-2) [sec], respectively. As shown in Fig. 3, the application receives events in the order of occurrence at T+10 [sec]. This example operates as a conservative synchronization algorithm. The delay for all users is the same at 10 [sec], which is set as $T_{\rm apl}$, and the delay difference between users is canceled. To ensure that users receive the processing results simultaneously, the server sends the results to each user after waiting for the maximum delay between the user and server minus the communication delay of the user. Each user receives the results with a delay of 10 [sec], the maximum delay between the user and the server. In other words, all users use the virtual processing system with the same 10 [sec] one-way delay.

Next, an example of the proposed scheme working as an optimistic synchronization algorithm is illustrated in Fig. 4 with $T_{\rm apl}$=7 and 3 [sec]. If the delay exceeds $T_{\rm apl}$, the event is passed to the application without waiting. If the application receives a past event, it performs a rollback process for the event order correction. In the example in Fig. 4(a), events a, b, and c wait 0 (-3=7-10), 2 (=7-5), and 5 (=7-2) [sec], respectively. The delay of 10 [sec] for event a is greater than 7 [sec] set as $T_{\rm apl}$, so the event is passed to the application without waiting. The application receives the event at T+7 [sec]. As event a is received 3 [sec] late, the application state is rolled back to 3 [sec] past state, and the result is corrected as if event a had been received.

Figure 4(b) is an example where $T_{\rm apl}$ is set to 3 [sec]. The application receives the event at T+3 [sec]. As events a and b are received 7 and 2 [sec] late, respectively, the application state is rollback to 7 and 2 [sec] past states, and the results are corrected as if events a and b had been received. In the conventional optimistic synchronization algorithm, a rollback process of up to 10 [sec] is required, representing the maximum delay [10]. However, the proposed scehme reduces the rollback time to 3 [sec] in Fig. 4(a) and 7 [sec] in Fig. 4(b). This contributes to reducing memory consumption for state retention, one of the drawbacks of the optimistic synchronization algorithm. To ensure that users receive the processing results simultaneously, the server sends the results to each user after waiting for the maximum delay between the user and server minus the communication delay of the user. Figures 4(a) and (b) show a virtual processing system in use with a one-way delay of 7 [sec] and 3 [sec], respectively.

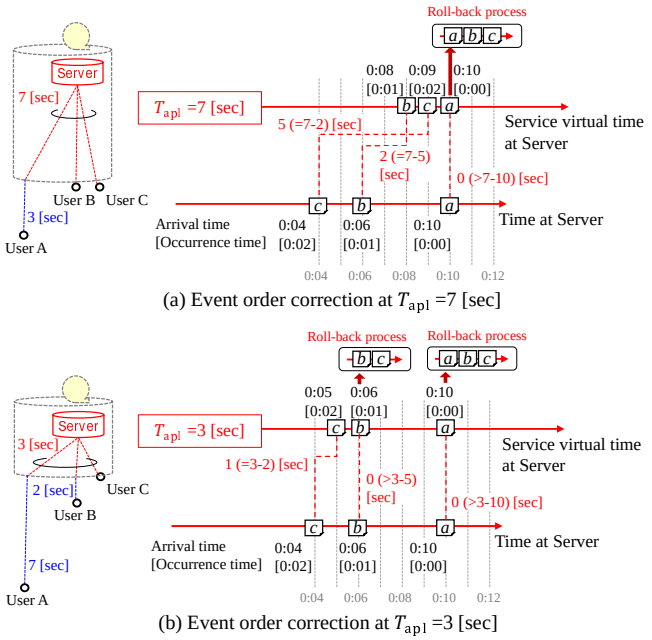


Fig. 4. Example of optimistic synchronization mode in proposed scheme.

TABLE I
ACTUAL DELAY AND MAXIMUM ROLLBACK TIME OF EACH SCHEME.

	Delay (D^{total})	Rollback Time (T^{roll})
Prop.	$T_{\text{apl}} + D_{\text{U}}^{\max}$	$D_{\text{US}}^{\max} - T_{\text{apl}}$
CSA	$D_{\text{US}}^{\max} + D_{\text{U}}^{\max}$	0
OSA	$D_{\text{U}}^{\max}$	$D_{\text{US}}^{\max}$

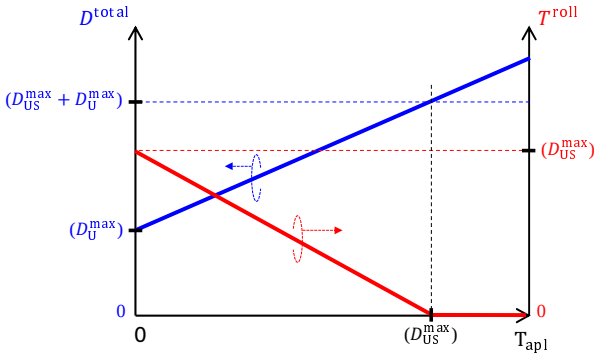


Fig. 5. Change in D^{total} and T^{roll} with the value of T_{apl} .

D. Notation of delay and memory consumption

We introduce the memory consumption for rollback and delay in the proposed scheme by comparing it with the conventional conservative and optimistic synchronization algorithms. In the proposed scheme, all user events are delivered to all servers. We denote the maximum delay between the users and servers as $D_{\text{US}}^{\max}$. In the example in Fig. 1, $D_{\text{US}}^{\max}$ is 11 [sec]. The result is returned from the accommodating server. We denote the largest delay between the user and the accommodating server as $D_{\text{U}}^{\max}$. In the example shown in Fig. 1, $D_{\text{U}}^{\max}$ is 3 [sec].

In the conservative synchronization algorithm, the delay when using the application (actual delay) is the sum of the maximum delay between the user and server (user to server), $D_{\text{US}}^{\max}$, plus the maximum delay between the user and accommodating server (server to user), $D_{\text{U}}^{\max}$. In [12], as the actual delay was treated $2D_{\text{U}}^{\max} + D_{\text{S}}^{\max}$ for calculation speed and simplicity, the actual delay was $D_{\text{US}}^{\max} + D_{\text{U}}^{\max}$. The optimistic synchronization algorithm does not queue the events at the receiving events, so only communication delays are considered. In the work [10], the actual delay is $2D_{\text{U}}^{\max}$, which is the round trip of the maximum user-server delay. If the time for rollback is $D_{\text{US}}^{\max}$, the processing result is corrected as if the events arrived $D_{\text{US}}^{\max}$ ago. Therefore, the maximum delay between the user and accommodating server (server to user), $D_{\text{U}}^{\max}$, is the actual delay. In the proposed scheme, the actual delay is $T_{\text{apl}} + D_{\text{U}}^{\max}$. This is the sum of the queuing delay before notifying the event to the application (user to server) and the maximum delay between the user and the accommodating server (server to user).

The memory consumption for a rollback is evaluated as the maximum rollback time. The longer the rollback, the more memory is required for state retention. The conservative synchronization algorithm has no rollback process, so there is no memory consumption. In optimistic synchronization, the maximum rollback time is the most late-arriving user event (maximum delay between the user and server; $D_{\text{US}}^{\max}$). In the proposed scheme, the delay of the user event that arrives latest to the application is the maximum rollback time, which is $D_{\text{US}}^{\max} - T_{\text{apl}}$. This is because events with a delay within T_{apl} are order-corrected, and events with a delay exceeding T_{apl} arrive at the application with a communication delay reduced by T_{apl}.

Table I shows each scheme's delay and memory consumption. The actual delay is denoted by D^{total}, and the amount of memory required for rollback is denoted by T^{roll}. In the proposed scheme, the actual delay, D^{total}, and the amount of memory required for rollback, T^{roll}, vary depending on T_{apl}. Figure 5 describes the change in D^{total} and T^{roll} depending on T_{apl}. The horizontal axis is T_{apl}, with the value increasing from left to right. The vertical axis on the left is D^{total}, and the blue graph shows the change in D^{total} depending on T_{apl}. As D^{total} is represented by $T_{\text{apl}} + D_{\text{U}}^{\max}$, as D^{total} increases, T_{apl} also increases. The vertical axis on the right is T^{roll}, and the red graph shows the change in T^{roll} depending on T_{apl}. Since T^{roll} is represented by $D_{\text{US}}^{\max} - T_{\text{apl}}$, T^{roll} decreases as T_{apl} increases. When T_{apl} is equal to $D_{\text{US}}^{\max}$, event order correction can be applied for all events, and no rollback processing is required on the application. When T_{apl} is less than $D_{\text{US}}^{\max}$, the proposed scheme works as a function to reduce the amount of memory for state retention for the optimistic synchronization algorithm. When T_{apl} is greater than $D_{\text{US}}^{\max}$, the proposed scheme operates for the event order correction for all events as the conservative synchronization algorithm. Thus, the proposed scheme can switch the synchronization mode with T_{apl} according to the application characteristics or network topology.

TABLE II
GIVEN PARAMETERS AND DECISION VARIABLES USED IN OPTIMIZATION PROBLEM.

Given Parameters	Description
V_{U}	Set of users
V_{S}	Set of servers
E_{U}	Set of links between user and server
E_{S}	Set of links between server and server
T_{apl}	Maximum queueing time
D_{cap}	Acceptable delay
M_i	Maximum accommodated users for server $i \in V_{\mathrm{S}}$
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**
U^{exc}	Number of excluded users
T^{roll}	Maximum rollback time
x_{pi}	$x_{pi} = 1$ if link $(p,i) \in E_{\mathrm{U}}$ is selected, $x_{pi} = 0$ if link $(p,i) \in E_{\mathrm{U}}$ is not selected or user $p \in V_{\mathrm{U}}$ is excluded
x_{ij}	$x_{ij} = 1$ if link $(i,j) \in E_{\mathrm{S}}$ is selected and $x_{ij} = 0$ otherwise
x'_{ij}	$x'_{ij} = 1$ if link $(i,j) \in E_{\mathrm{S}}$ is selected
y_i	$y_i = 1$ if server $i \in V_{\mathrm{S}}$ is selected
u_p	$u_p = 1$ if user $p \in V_{\mathrm{U}}$ is excluded, and $u_p = 0$ otherwise

The proposed scheme introduces D_{cap} as the acceptable delay for each application. For space-sharing applications, there is an acceptable delay in terms of quality of service and user operability. The service shall not be provided if the actual delay exceeds the acceptable delay. The distributed processing network must be designed to satisfy $D^{\mathrm{total}} \leq D_{\mathrm{cap}}$. In network design, users who cannot satisfy D_{cap} due to significant delay are excluded from the network. The proposed scheme designs the network to minimize the number of excluded users, U^{exc}, so that more users can be served (i.e., the maximum value of the actual delay is reduced).

E. Formulation

1) Parameters and variables: We present the formulation of the proposed scheme. The proposed scheme is formulated as a mixed-integer linear programming (MILP) problem to design the network. It minimizes the number of excluded users, U^{exc}, as the first-priority objective function and the maximum memory consumption for rollback as the second-priority objective function, under the condition of all constraints. By solving the MILP problem, the user-server and server-server connections are determined. Table II lists given parameters decision variables used in the optimization problem.

The network is modeled as an undirected graph (V, E), where V and E indicate sets of nodes and non-directional links, respectively. Let V_{U} denote the set of nodes representing, where $p \in V_{\mathrm{U}}$ denotes a node representing a user. Let V_{S} denote the set of nodes representing servers and $i \in V_{\mathrm{S}}$ denote a node representing a server. We have $V_{\mathrm{U}} \cup V_{\mathrm{S}} = V$ because there are only users and servers, and $V_{\mathrm{U}} \cap V_{\mathrm{S}} = \emptyset$ because there are no nodes that are users and servers. Let E_{U} denote the set of links between users and servers and $(p,i) \in E_{\mathrm{U}}$ denote the link between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$. Let E_{S} denote the set of links between servers and servers and $(i,j) \in E_{\mathrm{S}}$ denote the link between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}}$. We have $E_{\mathrm{U}} \cup E_{\mathrm{S}} = E$ as there are only user-server and server-server links, and $E_{\mathrm{U}} \cap E_{\mathrm{S}} = \emptyset$ because there are no links that are user-server and server-server links. Considering that users may be accommodated by a server other than the nearest one, $(p,i) \in E_{\mathrm{U}}$ is assumed to be set between every user $p \in V_{\mathrm{U}}$ and every server $i \in V_{\mathrm{S}}$. Similarly, $(i,j) \in E_{\mathrm{S}}$ is assumed to be set between all servers $i \in V_{\mathrm{S}}$ and $j \in V_{\mathrm{S}}$ (but $i \neq j$).

The given parameters are as follows. The delays of links $(p,i) \in E_{\mathrm{U}}$ and $(i,j) \in E_{\mathrm{S}}$ are denoted by d_{pi} and d_{ij}, respectively. This delay includes not only the transmission delay of optical fiber but also the delay of the network equipment on the transmission path. M_i is the maximum number of users that server $i \in V_{\mathrm{S}}$ can accommodate, which can be used for designing each server's capacity constraints. Y_{MAX} is the maximum number of servers in the network, which can be used for designing the network with a limited number of servers or applying a single server application (Y_{MAX} =1). The maximum queuing time for event order correction, as described in Section III-C2, is T_{apl}.

The decision variables are given below. x_{pi} is a binary variable for $(p,i) \in E_{\mathrm{U}}$, where x_{pi}=1 if link (p,i) is selected, and x_{pi}=0 otherwise. x_{ij} is a binary variable for $(i,j) \in E_{\mathrm{S}}$, where x_{ij}=1 if link (i,j) is selected. y_i is a binary variable for server $i \in V_{\mathrm{S}}$, where y_i=1 if at least one user selects server i. x_{ij} is a binary variable for $(i,j) \in E_{\mathrm{S}}$, where x_{ij}=1 if link (i,j) is selected, and x_{ij}=0 otherwise. x'_{ij} is a binary variable for $(i,j) \in E_{\mathrm{S}}$, where x'_{ij}=1 if link (i,j) is selected. x'_{ij} is forced to one if $y_i = y_j = 1$, but x'_{ij} is indeterminate otherwise (y_i=y_j=0, y_i=1 y_j=0, or y_i=0 y_j=1). x'_{ij} is introduced for a relaxation that potentially reduces computation time. u_p is a binary variable for $p \in V_{\mathrm{U}}$, where u_p=1 if user p is excluded, and u_p=0 otherwise. The number of excluded users and maximum rollback time are denoted as U^{exc} and T^{roll}, respectively.

2) Formulation and objective function: The proposed scheme is formulated with the following equations:

$$\text{Objective} \quad \min \ (U^{\mathrm{exc}} + \alpha T^{\mathrm{roll}}) \tag{1a}$$

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

$$\sum_{p \in V_{\mathrm{U}}} u_p \leq U^{\mathrm{exc}} \tag{1c}$$

$$\sum_{(p,i) \in E_{\mathrm{U}}} x_{pi} \leq M_i y_i, \forall i \in V_{\mathrm{s}} \tag{1d}$$

$$\sum_{i \in V_{\mathrm{S}}} y_i \leq Y_{\mathrm{MAX}} \tag{1e}$$

$$T_{\mathrm{apl}} + d_{pi} x_{pi} \leq D_{\mathrm{cap}}, \forall (p,i) \in E_{\mathrm{U}} \tag{1f}$$

$$(d_{pi} x_{pi} + d_{ij} x_{ij}) - T_{\mathrm{apl}} \leq T^{\mathrm{roll}},$$

$$\forall(p,i) \in E_{\mathrm{U}}, (i,j) \in E_{\mathrm{S}} \quad (1g)$$

$$x_{pi} \leq y_i, \forall p \in V_{\mathrm{U}}, i \in V_{\mathrm{S}} \quad (1h)$$

$$T^{\mathrm{roll}} \geq 0. \quad (1i)$$

$$y_i y_j = x_{ij}, \forall (i,j) \in E_{\mathrm{S}} \quad (1j)$$

$$x_{pi} \in \{0,1\}, \forall (p,i) \in E_{\mathrm{U}} \quad (1k)$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E_{\mathrm{S}} \quad (1l)$$

$$u_p \in \{0,1\}, \forall p \in V_{\mathrm{U}} \quad (1m)$$

$$y_i \in \{0,1\}, \forall i \in V_{\mathrm{S}} \quad (1n)$$

$$U^{\mathrm{exc}} \in \{0,1,2,3,\dots\}. \quad (1o)$$

The binary variable y_j appears in (1j) with index j, and its definition is provided in (1n), where the expression $y_i \in \{0,1\}, \forall i \in V_{\mathrm{S}}$, applies equivalently to y_j by substituting the index i with j.

First, we discuss the objective function. The formulated problem is a multi-objective problem, where the weighted sum method is applied with parameter α. In the problem, minimizing U^{exc} has a higher priority than minimizing T^{roll}. We denote U^{exc} as the first-priority objective function and T^{roll} as the second-priority objective function. α is set to satisfy $\alpha \leq \frac{1}{T^{\mathrm{roll}}_{\mathrm{UB}}+1}$, where $T^{\mathrm{UB}}_{\mathrm{roll}}$ represents the upper bound of T_{roll}. Under this condition, the contribution of the second term, αT^{roll}, is strictly less than 1, which is smaller than the minimum difference of U^{exe} (a non-negative integer). Therefore, αT^{roll} does not influence the minimization of the first term in U^{exe} when minimizing (1a). As a result, U^{exe} achieves its minimum value independently of the value of T_{roll}. Under this condition, T_{roll} is minimized as a secondary consideration. Consequently, U^{exe} is referred to as the first-priority objective function, while T_{roll} is regarded as the second-priority objective function. $T^{\mathrm{UB}}_{\mathrm{roll}}$ is determined based on the values of (d_{pi} + d_{ij}) from (1g) and (1i). Consequently, α is set by considering the maximum value of (d_{pi} + d_{ij}). This approach combines the two objective functions into a single scalar composite objective function by employing a weighted sum, as shown in (1a).

Next, we discuss the delays dealt with in the problem. As described in Section III-D, the the maximum delay of user-accommodating server, $D^{\max}_{\mathrm{U}}$, and maximum delay of user-all servers, $D^{\max}_{\mathrm{US}}$, are the upper bound on $d_{pi}x_{pi}$ and ($d_{pi}x_{pi} + d_{ij}x_{ij}$), respectively. Therefore, the actual delay, D^{total} (= T_{apl} + $D^{\max}_{\mathrm{U}}$), is the upper bound on $T_{\mathrm{apl}} + d_{pi}x_{pi}$.

In setting parameters, it must be carefully set the value of T_{apl}. T^{roll} must be a non-negative continuous variable. To maintain this condition, T_{apl} must not be set greater than the maximum value of ($d_{pi}x_{pi}$ + $d_{ij}x_{ij}$). When T_{apl} is greater than the maximum value of ($d_{pi}x_{pi}$ + $d_{ij}x_{ij}$), the delay increases even though T^{roll} does not become smaller (the rollback time does not become shorter), as shown in Fig. 5.

3) Constraints: Equation (1b) indicates that the user who is not excluded selects one server from several available servers. Equation (1c) indicates that the number of excluded users does not exceed U^{exc}. Equation (1d) indicates that the number of accommodated users in server $i \in V_{\mathrm{S}}$ does not exceed M_i at selected server. Equation (1e) indicates that the maximum number of selected servers in the network does not exceed Y_{MAX}. Equation (1f) indicates that the actual delay, D^{total} (=$T_{\mathrm{apl}} + D^{\max}_{\mathrm{U}}$), does not exceed the acceptable delay of the service, D_{cap}. D^{total} is the upper bound on T_{apl} + $d_{pi}x_{pi}$. Equation (1g) indicates that the maximum rollback time, T^{roll}, does not exceed $D^{\max}_{\mathrm{US}} - T_{\mathrm{apl}}$. D^{US} is the upper bound on ($d_{pi}x_{pi} + d_{ij}x_{ij}$). Equation (1h) indicates that y_i=1 if server $i \in V_{\mathrm{S}}$ is selected by at least one user. We note that the value of y_i can be arbitrary if server $i \in V_{\mathrm{S}}$ is not selected by any user, which does not affect the optimal objective value of (1a) to be minimized. Equation (1i) indicates that T^{roll} is a non-negative continuous variable. Equation (1j) indicates that $x_{ij} = 1$ if $y_i = 1$ and $y_j = 1$. Equations (1k)–(1n) indicate that x_{pi}, x_{ij}, y_i, and u_p are binary variables. Equation (1o) indicates that U^{exc} is a non-negative integer.

4) Linearization of the problem as MILP: We linearize (1a)–(1o) as an MILP problem. x_{ij}, y_i, and y_j are binary variables, $y_i y_j = x_{ij}$ expressed in (1j) is linearized [25], [26], and we formulate (1a)–(1o) as the following MILP problem:

$$\text{Objective} \quad \min \ (U^{\mathrm{exc}} + \alpha T^{\mathrm{roll}}) \quad (2a)$$

$$\text{s.t.} \quad y_i + y_j - 1 \leq x_{ij}, \forall (i,j) \in E_{\mathrm{S}} \quad (2b)$$

$$x_{ij} \leq y_i, \forall i \in V_{\mathrm{S}}, (i,j) \in E_{\mathrm{S}} \quad (2c)$$

$$x_{ij} \leq y_j, \forall j \in V_{\mathrm{S}}, (i,j) \in E_{\mathrm{S}}. \quad (2d)$$

$$(1b) - (1i), (1k) - (1o). \quad (2e)$$

We introduce a binary variable x'_{ij} for $(i,j) \in E_{\mathrm{S}}$ for a relaxation that potentially reduces computation time. The variable x'_{ij} is set to 1 if $y_i = 1$ and $y_j = 1$, while its value is left unspecified otherwise. By replacing x_{ij} with x'_{ij}, we reformulate (2a)–(2e) as follows:

$$\text{Objective} \quad \min \ (U^{\mathrm{exc}} + \alpha T^{\mathrm{roll}}) \quad (3a)$$

$$\text{s.t.} \quad (d_{pi}x_{pi} + d_{ij}x'_{ij}) - T_{\mathrm{apl}} \leq T^{\mathrm{roll}}, \quad \forall (p,i) \in E_{\mathrm{U}}, (i,j) \in E_{\mathrm{S}} \quad (3b)$$

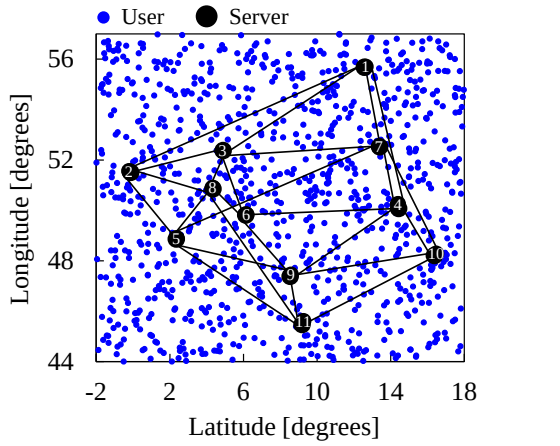
$$y_i + y_j - 1 \leq x'_{ij}, \forall (i,j) \in E_{\mathrm{S}} \quad (3c)$$

$$x'_{ij} \in \{0,1\}, \forall (i,j) \in E_{\mathrm{S}} \quad (3d)$$

$$(1b) - (1f), (1h), (1i), (1k), \quad (1m) - (1o). \quad (3e)$$

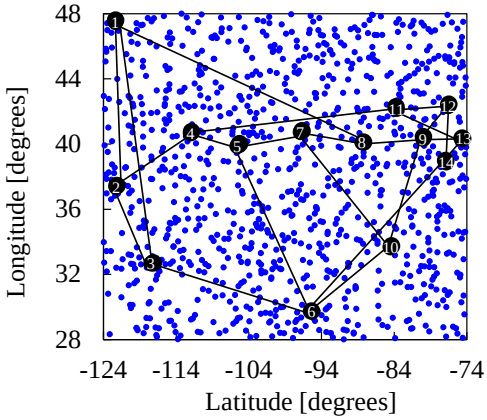
We note that the value of x'_{ij} can be arbitrary when $y_i = 0$ or $y_j = 0$, but this does not affect the optimal objective value of (3a). This is because we minimize the objective in (3a) subject to (3b), which forces $x'_{ij} = 0$ whenever x'_{ij} contributes to the objective and either $y_i = 0$ or $y_j = 0$. Modern optimization solvers such as CPLEX, Gurobi, and SCIP [27]–[29] appropriately handle such situations under the integrality constraints that x'_{ij}, y_i, and y_j are binary variables.

5) Numbers of variables and constraints: The numbers of variables and that of constraints in (3a)–(3e) are analyzed in the following. Binary variables x_{pi}, x'_{ij}, u_p, and y_i appear $|E_{\mathrm{U}}|$, $|E_{\mathrm{S}}|$, $|V_{\mathrm{U}}|$, and $|V_{\mathrm{S}}|$ times, respectively. Variables U^{exc} and T^{roll} appear two times, three times, respectively. Therefore, the total number of variables in (3a)–(3e) is $|E_{\mathrm{U}}| + |E_{\mathrm{S}}| + |V_{\mathrm{U}}| + V_{\mathrm{S}}| + 5$. The number of variables in the MILP problem is $O(|E_{\mathrm{U}}| + |E_{\mathrm{S}}| + |V_{\mathrm{U}}| + |V_{\mathrm{S}}|)$. The numbers



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

(a) COST



1	Seattle	6	Houston	11	Ann Arbor
2	Palo Alto	7	Lincon	12	Ithaca
3	San Diego	8	Champaign	13	Princeton
4	Salt Lake	9	Pittsburgh	14	College Pk
5	Boulder	10	Atlanta		

(b) NSFnet

Fig. 6. Location of users and servers.

of constraints in (1b), (1c), (1d), (1e), (1f), (1h), (1i), (3b), and (3c) are $|V_{\rm U}|$, 1, $|V_{\rm S}|$, 1, $|E_{\rm U}|$, $|V_{\rm U}||V_{\rm S}|$, 1, $|E_{\rm U}||E_{\rm S}|$, and $|E_{\rm S}|$, respectively. Therefore, the total number of constraints in (3a)–(3e) is $|V_{\rm U}|+|V_{\rm S}|+|E_{\rm U}|+|V_{\rm U}||V_{\rm S}|+|E_{\rm U}||E_{\rm S}|+|E_{\rm S}|$. The number of constraints in the MILP problem is $O(|V_{\rm U}|+|V_{\rm S}|+|E_{\rm U}|+|E_{\rm S}|+|V_{\rm U}||V_{\rm S}|+|E_{\rm U}||E_{\rm S}|)$. In the worst case, we have $|E_{\rm U}|$=$|V_{\rm U}||V_{\rm S}|$ and $|E_{\rm S}|$=$|V_{\rm S}|^2$. Thus, the number of constraints can be expressed as $O(|V_{\rm U}|+|V_{\rm S}|+|V_{\rm U}||V_{\rm S}|+|V_{\rm S}|^2+|V_{\rm U}||V_{\rm S}|^3)$.

IV. Evaluation

A. Methodology of evaluation

The environment for solving the MILP problem is a generally available general-purpose computer and a solver. We use IBM ILOG CPLEX Optimization Studio (CPLEX) as the solver [27]. CPLEX implements various algorithms such as Branch and Bound and Branch and Cut [30]. In this evaluation, the algorithm is used with the default settings in CPLEX, and the solution is generated using the sifting algorithm. The parameters are left at the default values. The main parameters are as follows. The value of the feasibility tolerance is 1×10^{-6}. The value of the integrality tolerance is 1×10^{-5}. The value of the relative gap tolerance between the best integer objective and the objective of the best node remaining is 1×10^{-4}. The value of the absolute gap tolerance is 1×10^{-6}. The value of the lower cutoff is -1×10^{75}; the default value of the upper cutoff is 1×10^{75}. For the evaluation, these values affect the determination of $T^{\rm roll}$. The value of $T^{\rm roll}$ is used to calculate the amount of memory for the rollback process at the application. Since memory is usually allocated with a margin of error of at least a few tens of percent, this level of accuracy is acceptable.

We use (3a)–(3e) for obtaining an optimal solution. In an MILP approach, the computational time increases with the problem size, although it can obtain an optimal solution. Therefore, in evaluating the proposed scheme, we evaluate the computation time in addition to the basic performance, i.e., delay characteristics and memory consumption.

B. Conditions of evaluation

We evaluate the proposed scheme on two different networks: COST [8] and NSFnet [9], as shown in Figs. 6(a) and (b), respectively. We assume that the available servers are located at COST and NSFnet node locations and that the links between the servers have the same configuration as COST and NSFnet links. Server-server links are assumed to be a logical full-mesh configuration, connected by the shortest route on the COST and NSFnet links, even if there are no directly connected links. For example, the link between servers A-C is assumed to exist, connected via the node located server B, even if the only direct links are between servers A-B and B-C. Users are assumed to be randomly distributed in a rectangular area, and each user is assumed to have a direct link to all servers. This is because all servers may be available for all users. The links between users and servers are assumed to be three times the linear distance on the coordinate axes. This assumption is made because metro networks often take a ring topology [31], and many devices are likely to be present in the access network. In practice, the delay is estimated using the time-stamp information from received frames. For simplicity, the delay is assumed to be proportional to the connection distance. The distance between two points is estimated as a straight line distance obtained from latitude and longitude, and the delay is calculated as 5 [ms] per kilometer based on the transmission delay of optical fibers.

C. Performance evaluation without server capacity constraints

We evaluate the dependence of $T_{\rm apl}$ on the number of excluded users, $U^{\rm exc}$, and the amount of memory consumed (maximum rollback time), $T^{\rm roll}$. $T_{\rm apl}$ is a time parameter related to delay fairness. The delay difference between users in an application is decreased as $T_{\rm apl}$ increases. All servers are set to M_i=1000; there are no constraints on the number

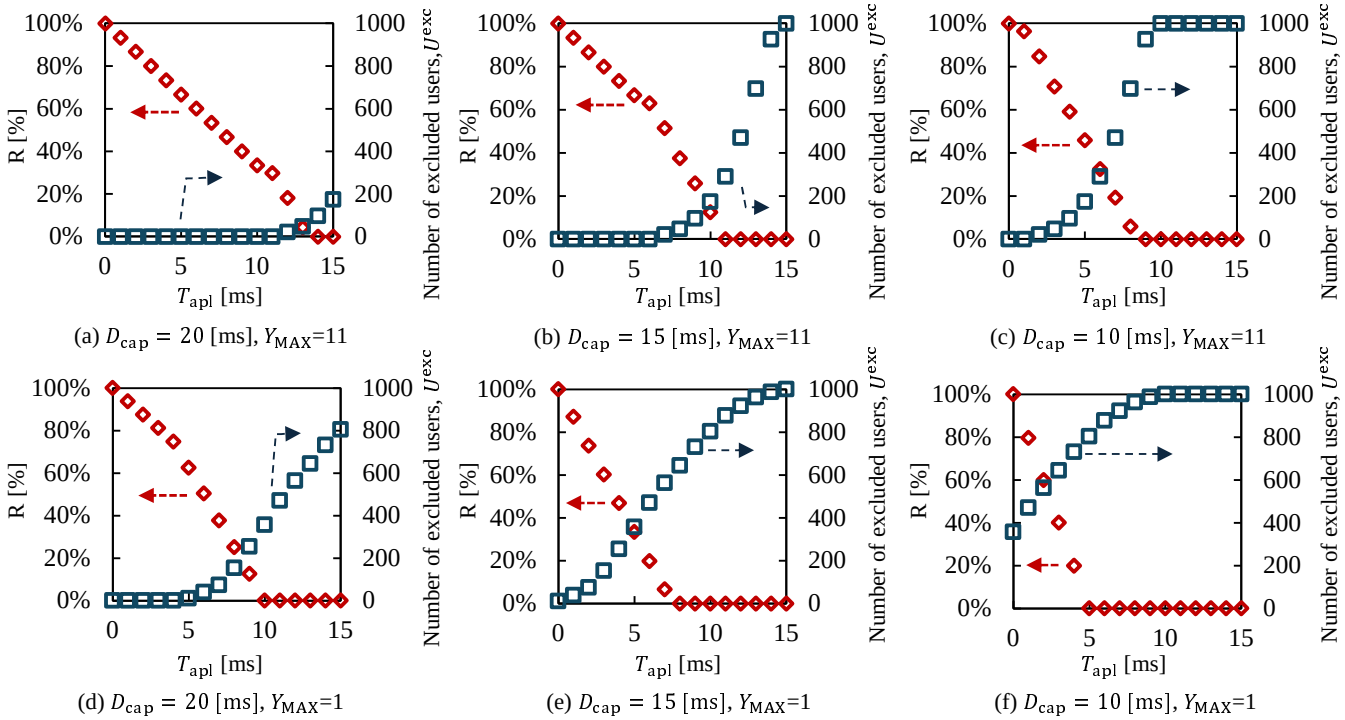


Fig. 7. T_{apl} dependence of excluded users, U^{exc}, and and memory reduction ratio, R, in COST.

of accommodating users. R denotes the memory reduction ratio of the proposed scheme. R is expressed as a ratio of the maximum rollback time of the proposed scheme to the maximum rollback time of the conventional optimistic synchronization algorithm, i.e., R is given by:

$$R = \frac{\text{Maximum rollback time in proposed scheme}}{\text{Maximum rollback time in optimistic synchronization}}.$$

Table I presents the maximum rollback time of the proposed scheme and the conventional optimistic synchronization algorithm.

Figure 7 shows the results evaluated in COST. Figures 7(a), (b), and (c) present the results with Y_{MAX}=11, i.e., all servers available for distributed processing. Figures 7(d), (e), and (f) present the results with Y_{MAX}=1 for a centralized processing configuration with only one server. We consider D_{cap} = 20 [ms] in Figs. 7(a) and (d), D_{cap} = 15 in Figs. 7(b) and (e), and D_{cap} = 10 [ms] in Fig. 7(c). The acceptable delay constraints become progressively tighter. The common trend of these results is that as T_{apl} increases, the memory reduction ratio, R, decreases and the number of excluded users, U^{exc}, increases. This is because T^{roll} decreases as T_{apl} increases from (3b). The proposed scheme's event order correction reduces the application's rollback time. $D_{\mathrm{U}}^{\mathrm{max}}$ decreases as T_{apl} increases from (1f). Furthermore, as $D_{\mathrm{U}}^{\mathrm{max}}$ becomes smaller, users with large d_{pi} are excluded. For these reasons, as T_{apl} increases, the memory reduction ratio R decreases and the number of excluded users U^{exc} increases.

Next, we discuss the effects of changes in D_{cap}. As shown in Figs. 7(a) and (b), the memory reduction ratio R is reduced proportionally as T_{apl} is reduced under the condition that no user is excluded by the constraint of D_{cap}. This is because T^{roll} decreases as T_{apl} increases from (3b). Meanwhile, in Fig. 7(b), as U^{exc} increases, the memory reduction ratio R changes from a proportional relationship with T_{apl}. This is because users with large d_{pi} who determine $D_{\mathrm{U}}^{\mathrm{max}}$ are excluded by the constraints of D_{cap} by increasing T_{apl}. The proportional relationship between R and T_{apl} is broken because (i) the user who determines $D_{\mathrm{US}}^{\mathrm{max}}$ is excluded and (ii) $D_{\mathrm{US}}^{\mathrm{max}}$ is changed. However, as the user who determines $D_{\mathrm{U}}^{\mathrm{max}}$ is not necessarily the user who determines $D_{\mathrm{US}}^{\mathrm{max}}$, $D_{\mathrm{US}}^{\mathrm{max}}$ may not change even if the number of excluded users increases. It depends on the network topology to determine which case applies.

We compare the distributed configuration (Figs. 7(a)–(c) with $Y_{\mathrm{MAX}} = 11$) and centralized configuration (Figs. 7(d)–(f) with $Y_{\mathrm{MAX}} = 1$). The number of excluded users in the latter is more significant than that of the former when T_{apl} is the same. Under the condition of D_{cap}=20 [ms], the maximum value of T_{apl} at which the number of excluded users becomes zero is 11 for Y_{MAX}=11 and four for Y_{MAX}=1. This result shows that distributed processing reduces the delay compared to centralized processing. These results indicate that the distributed configuration has better delay characteristics in actual delay because $D_{\mathrm{US}}^{\mathrm{max}}$ can be smaller. This is due to the communication model described in Fig. 1. In this model, each distributed processing server multicasts the events of its accommodated users to other servers. As shown in Fig. 1, the maximum round trip time of the distributed configuration is 14 [sec] ($D_{\mathrm{US}}^{\mathrm{max}}$=11 and $D_{\mathrm{U}}^{\mathrm{max}}$=3). However, if the centralized processing is configured on server 2, the maximum round trip time is 16 [sec] ($D_{\mathrm{US}}^{\mathrm{max}}$=$D_{\mathrm{U}}^{\mathrm{max}}$=8).

We summarize the above results as follows. The proposed scheme can reduce the memory consumption for rollback by

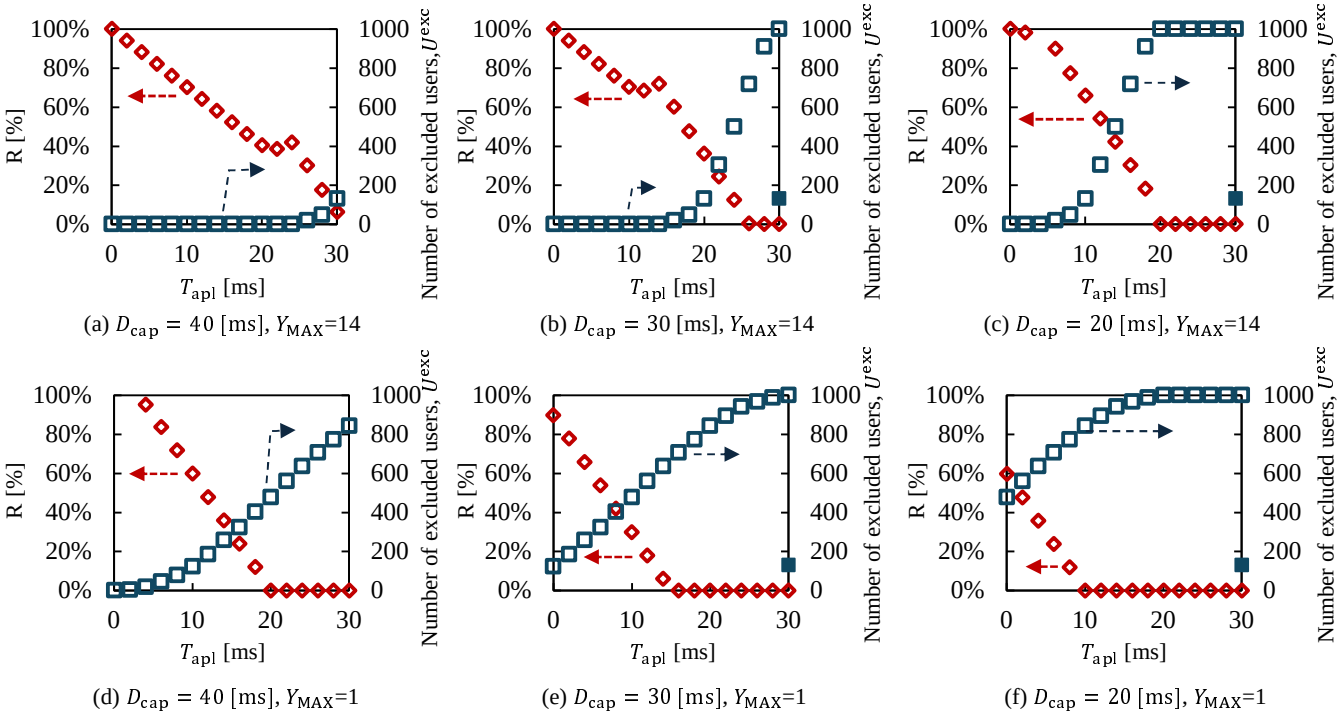


Fig. 8. T_{apl} dependence of excluded users, U^{exc}, and memory reduction ratio, R, in NSFnet.

increasing T_{apl}. The users whose delay is smaller than T_{apl} also have a delay of T_{apl}, but they are satisfied with the acceptable delay of the application. As T_{apl} increases, R becomes zero and the application does not need the rollback process, i.e., all events are notified to the application in the occurrence order. In other words, the proposed scheme operates as a function of the conservative synchronization algorithm. Thus, depending on the setting of T_{apl}, it can act as a function to reduce the memory consumption for rollback for applications supporting the optimistic synchronization algorithm or as a function to realize the conservative synchronization algorithm. The value of T_{apl} should be set according to the network topology and quality requirements of the application.

We discuss the results of evaluation across different networks. Figure 8 shows the dependence of T_{apl} on the number of excluded users, U^{exc}, and the amount of memory consumed, T^{roll}, in NSFnet. Figures 8(a), (b), and (c) show the results with Y_{MAX}=14, i.e., all servers available for distributed processing. Figures 8(d), (e), and (f) show the results with Y_{MAX}=1 for a centralized processing configuration with only one server. We consider D_{cap} = 40 [ms] in Figs. 8(a) and (d), D_{cap} = 30 [ms] in Figs. 8(b) and (e), and D_{cap} = 20 [ms] in Figs. 8(c) and (f). The acceptable delay constraints become progressively tighter. As with the evaluation in COST, as T_{apl} increase, the memory reduction ratio R decreases and the number of excluded users U^{exc} increases. Under the condition of D_{cap}=30 [ms], the maximum value of T_{apl} at which the number of excluded users becomes under 200 is 20 for Y_{MAX}=14 and two for Y_{MAX}=1. This result shows that distributed processing reduces the delay compared to centralized processing.

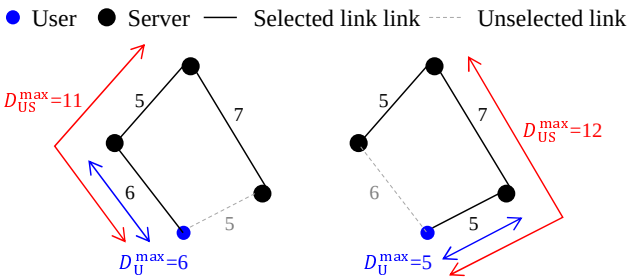
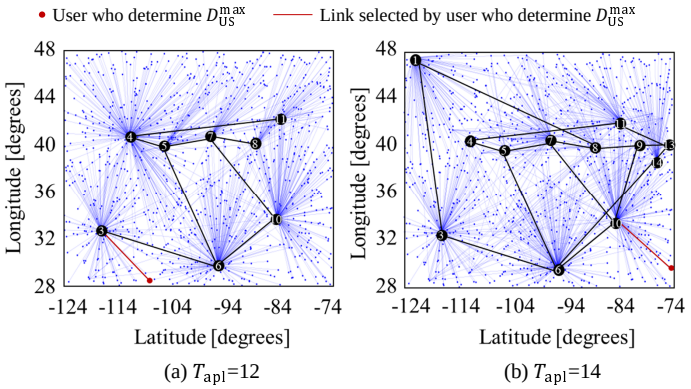


Fig. 9. Example of D_{US}^{max} increasing due to change of selected server.

A different trend from the COST evaluation is that in Figs. 8(a) and (b), the memory reduction ratio R is temporarily worse with an increase in T_{apl}. This is because the user and server combination that determines D_{US}^{max} changes. Figure 9 shows an example where an increase in T_{apl} causes the selected server to change. As shown in Fig. 9, the constraint on D_{cap} causes the selected server to change; as a result, D_{U}^{max} becomes smaller, and D_{US}^{max} becomes larger. Figures 10(a) and (b) show the designed networks at the condition of Fig. 8(b) in T_{apl}=12 and 14. These results show that the user-server pair with D_{U}^{max} is changed and T_{roll} is worsened by the D_{cap} constraint. However, as shown in Figs. 8(d), (e), and (f), no temporary deterioration of the R is observed in one server configuration because no selection server switching has occurred.

Similar to COST, the number of excluded users in distributed processing is reduced compared to that of the cen-



(a) T_{apl}=12 (b) T_{apl}=14

Fig. 10. Designed network in NSFnet at D_{cap}=30 [ms], Y_{MAX}=14, and M_i=1000.

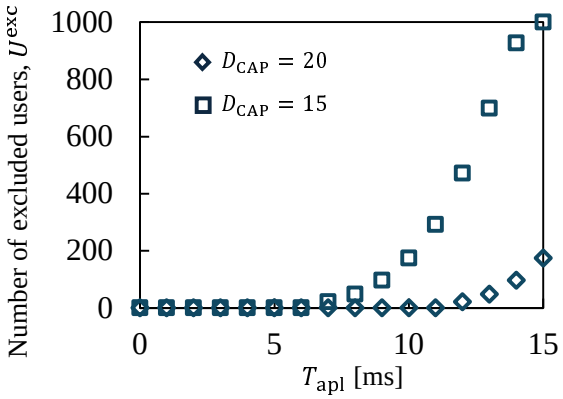
tralized processing configuration. For emample, in D_{cap}=40 and T_{apl}=10, U^{exc} with Y_{MAX}=11 and Y_{MAX}=1 are 0 and 123, as shown in Figs. 8(a) and (d), respectively. In D_{cap}=30 and T_{apl}=10, U^{exc} with Y_{MAX}=11 and Y_{MAX}=1 are zero and 477, as shown in Figs. 8(b) and (e), respectively. These results show that in NSFnet, as in the evaluation of COST, the proposed scheme can act as a function to reduce the memory consumption for rollback for applications supporting the optimistic synchronization algorithm or as a function to realize the conservative synchronization algorithm.

D. Performance evaluation with server capacity constraints

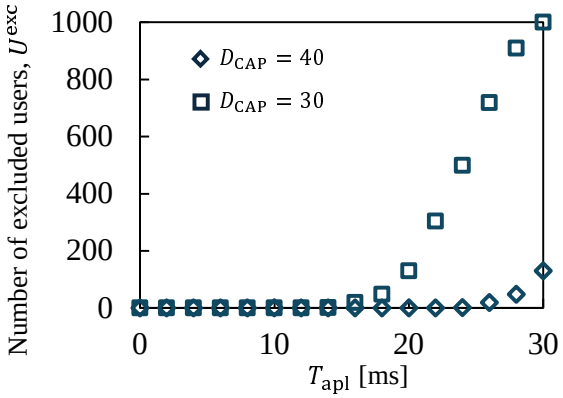
We conduct performance evaluation under conditions where the number of accommodating users is limited by server processing performance. The condition is the same as Section IV-C except for M_i=100. Figures 11(a) and (b) show the dependence of T_{apl} on the number of excluded users, U^{exc}, in COST and NSFnet. The results in Figs. 11(a) and (b) are identical to those under no server capacity constraint regardless of D_{cap}; Figs. 11(a) and (b) present the results corresponding to those of Figs. 7(a) and (b) in COST and those of Figs. 8(a) and (b) in NSFnet, respectively. The value of D_U^{max}, which determines the number of excluded users, is determined by the maximum delay between the user and the server. In other words, the deterioration of the number of excluded users does not occur except for the two cases; one is that M_i of the server with the maximum delay user is set to zero; the other is that D_U^{max} increases due to the relationships of the users and the accommodating servers are changed due to the constraints of M_i.

E. Computation time

We evaluate the computation time to determine the network configuration in the proposed scheme. Table III shows the compuation time at T_{apl}=5 and 10 under the conditions of Fig. 7(b), and T_{apl}=10 and 20 in the condition of Fig. 8(b). Each computation time is an average of five times. It takes less than nine [sec] at most, which is considered acceptable for a preparation time to determine the network configuration for pre-planned service.



(a) COST, M_i=100 (i=1−11), Y_{MAX}=11



(b) NSFnet, M_i=100 (i=1−14), Y_{MAX}=14

Fig. 11. T_{apl} dependence of excluded users, U^{exc} under constraint of M_i.

TABLE III
AVERAGE COMPUTATION TIME (MAX, MIN) FOR 1000 USERS.

	Condition	Time [sec]
COST	Y_{MAX}=11, D_{cap}=15 [ms], T_{apl}=5	2.53 (2.62, 2.49)
	Y_{MAX}=11, D_{cap}=15 [ms], T_{apl}=10	0.15 (0.16, 0.15)
NSFnet	Y_{MAX}=14, D_{cap}=30 [ms], T_{apl}=10	8.76 (9.27, 8.39)
	Y_{MAX}=14, D_{cap}=30 [ms], T_{apl}=20	0.28 (0.29, 0.28)

Next, we investigate the impact of increasing the number of users. In general, an approach using MILP can yield an optimal solution, but as the scale of the problem increases, there is a concern that the computation time increases. Figure 12 shows the computation time when the number of users varied from 100 to 10000 at the condition of COST in Fig. 7(b) and Fig. 11(a). For 10000 users with T_{apl}=5, M_i=1000 (all $i \in V_S$), the computation time is 960 [sec]. This is acceptable if the user is determined in advance and the service is provided in a planned manner, but if the service starts as soon as the user participates, the waiting time may be unacceptable. Although it is rare for a service to start as soon as 10000 users participate, speeding up computation time will be a challenge for future consideration.

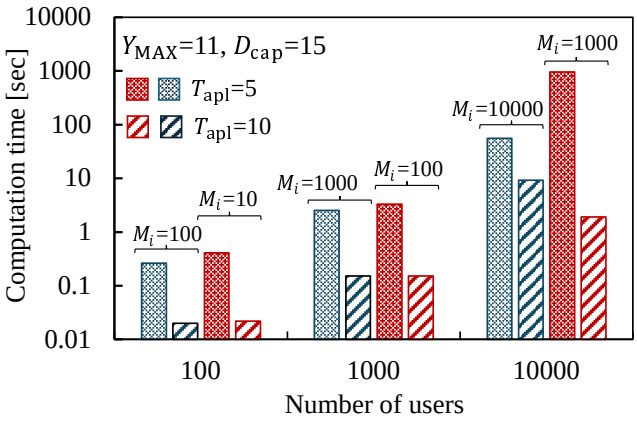


Fig. 12. Computation time depending on the number of users.

Since the original problem (1a)–(1o) is a nonconvex mixed integer nonlinear, the computation time depends on the strictness of the convex relaxation. Since the convex relaxation depends on the bounds of the variables contained in the nonconvex nonlinear terms (i.e., y_i and y_j), there is a concern that the computation time will increase with the number of servers and server-server links.

V. Conclusion

In this paper, we proposed a distributed processing network design scheme for space-sharing applications. The scheme is intended to be used in a virtual application processing platform with multiple application servers distributed over a wide-area network. The proposed scheme introduced $T_{\rm apl}$, the maximum queuing time for event order correction, as a given parameter related to delay fairness, and events within $T_{\rm apl}$ delay are sorted in order of occurrence. Setting $T_{\rm apl}$ to the maximum delay between users and servers, the proposed scheme works as a conservative synchronization. Setting $T_{\rm apl}$ to smaller than the maximum delay, the proposed scheme works as a function to reduce the maximum rollback time in an optimistic synchronization algorithm. Thus, the proposed scheme can change the mode of operation from conservative synchronization to optimistic synchronization depending on $T_{\rm apl}$. The proposed scheme determines the user-server and server-server network configuration under the constraints of the acceptable delay in application, server capacity, and number of servers in the network. The novelty of the proposed scheme is its generic usability in various applications, in that it is possible to switch the mode to conservative or optimistic synchronization algorithms by setting $T_{\rm apl}$.

The proposed scheme was evaluated on two network topologies: COST and NSFnet. The evaluation results indicate that, depending on the setting of $T_{\rm apl}$, the proposed scheme reduces the amount of memory consumed for rollback for optimistic synchronization-based applications or as a function that enables the conservative synchronization algorithm. The evaluation results indicate that, depending on the setting of $T_{\rm apl}$, the proposed scheme reduces the amount of memory consumed for rollback for optimistic synchronization-based applications or as a function that enables conservative synchronization algorithms. In the proposed scheme, the delay performance improves when distributed processing is used. The delay of distributed processing is reduced by 1.3 times compared to that of centralized processing in COST when the acceptable delay is 10 [ms], and the delay of distributed processing is reduced by 2.5 times compared to that of centralized processing in NSFnet when the delay acceptable delay is 30 [ms]. The computation time of the proposed scheme is within nine [sec] for 1000 users, which is acceptable for a preparation time to determine the network configuration for pre-planned service. These results indicate that the proposed scheme is effective in a virtual processing processing platform for space-sharing applications.

To address networks with dynamically changing traffic and users over time, such as mobile networks or successive participation scenarios where users continuously enter and exit the network, issues such as calculating maximum delay and dynamically changing queuing time on the server must be further considered. If the maximum value of the varying delay can be estimated, a possible solution is to design with the maximum delay. In addition, in the proposed scheme, all servers process the same events in the same order simultaneously, so the results on all servers are assumed to be consistent. This assumption is not an issue in applications where data is highly independent. However, in applications with strict constraints on data consistency, we may need to consider data consistency and the reliability of the network and servers. Analyzing constraints for adapting the proposed scheme depending on each application's characteristics and evaluating the reliability of the network and servers will be necessary. Network redundancy [32] and application data redundancy [33] have been extensively studied in conservative synchronization algorithms, and incorporating these concepts into the proposed scheme is a promising direction. Further exploration of these aspects in relation to the proposed scheme is left for future work. Furthermore, the computation time depends on the bounds of the variables contained in the nonconvex nonlinear terms (i.e., y_i and y_j), there is a concern that the computation time will increase with the number of servers and server-server links. Evaluation of these issues, including the target network selection, is left as part of future works.

Acknowledgement

This work was supported by ROIS NII Open Collaborative Research 251S1-22586 and JSPS KAKENHI JP23K16867. The authors would like to thank the 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.docomo.ne.jp/english/info/media_center/pr/2020/0318_00.html, online; accessed 5 April 2024.
[2] "AWS for the Edge," https://aws.amazon.com/edge/services/?nc1=h_ls, online; accessed 5 April 2024.
[3] "Azure IoT Edge," https://azure.microsoft.com/en-us/products/iot-edge, online; accessed 5 April 2024.

[4] 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.
[5] R. M. Fujimoto, *Parallel and Distributed Simulation System.* New York, NY, USA: Wiley, 2000.
[6] D. R. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 7, no. 3, pp. 404–425, 1985.
[7] 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 Conference on Communication*, vol. 1, pp. 6, 2005.
[8] M. O'mahony, "Ultrahigh capacity optical transmission network: European research project cost 239," *Information, Telecommunications, Automata Journal*, vol. 12, pp. 33–45, 1993.
[9] K. C. Claffy, G. C. Polyzos, and H. W. Braun, "Traffic characteristics of the T1 NSFNET backbone," *IEEE Infocom'93 The Conference on Computer Communications, Proceedings*, pp. 885–892, 1993.
[10] A. Kawabata, B. C. Chatterjee, S. Ba, and E. Oki, "An optimistic synchronization based optimal server selection scheme for delay sensitive communication services," *IEICE Transactions on Communications.*, vol. 104, no. 10, pp 1277–1287, 2021.
[11] S. Hiragi, B. C. Chatterjee, E. Oki, and A. Kawabata, "A distributed processing communication scheme for real-time applications over wide-area networks" *IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 400–40, 2024.
[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. Ferscha and J. Luthi, "Estimating rollback overhead for optimism control in Time Warp," *Proceedings of Simulation Symposium*, pp. 2–12, IEEE, 1995.
[14] D. R. Jefferson, and D. B. Peter, "Virtual time III, Part 1: Unified virtual time synchronization for parallel discrete event simulation," *ACM Transactions on Modeling and Computer Simulation*, vol. 32, no. 4, pp. 1–29, 2023.
[15] D. R. Jefferson, and D. B. Peter, "Virtual time III, Part 2: Combining conservative and optimistic synchronization," *ACM Transactions on Modeling and Computer Simulation*, vol. 32, no. 4, pp. 1–21, 2023.
[16] "GGPO Rollback Networking SDK," https://github.com/pond3r/ggpo, online; accessed 5 April 2024.
[17] T. Ishioka, T. Fukui, T. Fujiwara, S. Narikawa, T. Fujihashi, S. Saruwatari, and T. Watanabe, "Pattern reduction for low-traffic speculative video transmission in cloud gaming system," *IEEE Access*, 2024.
[18] E. Gupta, P. Goyal, I. Marinos, C. Zhao, R. Mittal, and R. Chandra, "DBO: Fairness for cloud-hosted financial exchanges" *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 550–563, 2023.
[19] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
[20] R. B. Haim and O. Rottenstreich, "Low-latency and reliable virtual network function placement in edge clouds," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2172–2185, 2023.
[21] C. C. Lin, Y. Chiang and H. Y. Wei, "Multi-service edge computing management with multi-stage coalition game task offloading," *IEEE Transactions on Network and Service Management*, IEEE, 2024.
[22] D. Mills, U. Delaware, J. Martin, J. Burbank, and W. Kasch, "RFC 5905 - Network time protocol version 4: protocol and algorithms specification," *Internet Engineering Task Force (IETF)*, 2010.
[23] J. C. Eidson, M. Fischer, and J. White, "IEEE-1588 Standard for a precision clock synchronization protocol for networked measurement and control systems," *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, pp. 243–254, 2002.
[24] "Global IP network," https://www.ntt.com/en/services/network/gin/sla.html, online; accessed 5 April 2024.
[25] Y. Hirano, F. He, T. Sato, and E. Oki, "Backup network design against multiple link failures to avoid link capacity overestimation," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1254–1267, 2020.
[26] H. Taka, T. Inoue, and E. Oki, "Twisted and folded clos-network design model with two-step blocking probability cuarantee" *IEEE Networking Letters*, vol. 6, no. 1, pp. 60-64, 2024.
[27] "CPLEX optimization studio," https://www.ibm.com/products/ilog-cplex-optimization-studio?mhsrc=ibmsearch_a&mhq=cplex, online; accessed 5 April 2024.
[28] "Gurobi Optimization," https://www.gurobi.com/, online; accessed 12 April 2025.
[29] "Solving Constraint Integer Programs (SCIP)," https://www.scipopt.org/, online; accessed 12 April 2025.
[30] R. E. Bixby, "A brief history of linear and mixed-integer programming computation," *Documenta Mathematica*, pp. 107–121, 2012.
[31] S. Matsuoka, "Ultrahigh-speed ultrahigh-capacity transport network technology for cost-effective core and metro networks," *NTT Technical Review*, vol. 9, no. 8, 2011.
[32] A. Kawabata, B. C. Chatterjee, and E. Oki, "MHND: Multi-Homing network design model for delay sensitive applications," *IEICE Transactions on Communications*, vol. E106-B, no. 11, pp. 1143–1153, 2023.
[33] A. Kawabata, B. C. Chatterjee, and E. Oki, "CMND: Consistent-aware multi-server network design model for delay-sensitive applications," *IEICE Transactions on Communications*, vol. E107-B, no. 3, pp. 321–329, 2024.

Akio Kawabata is a Professor at Toyohashi University of Technology, Aichi, 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. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2022. He has been a Professor at Toyohashi University of Technology since 2022. He served as a senior manager of R&D Department at NTT East from 2011 until 2014. His research interests include distributed systems, network architecture, and switching systems.

Sanetora Hiragi was a student in a master's course at Toyohashi University of Technology, Aichi, Japan. He received the B.E. and M.E. degrees from Toyohashi University of Technology, Aichi, Japan, in 2023 and 2025, respectively. His research interests include distributed systems and network architecture.

Bijoy Chand Chatterjee is an Associate Professor at South Asian University (SAU), New Delhi, and an Adjunct Professor at the Indraprastha Institute of Information Technology Delhi (IIITD), New Delhi, India. Before joining SAU, he was with IIITD, Norwegian University of Science and Technology, Trondheim, Norway, and The University of Electro-Communications, Tokyo, Japan. His research interests include optical networks, QoS-aware protocols, optimization, and routing. He is a senior member of IEEE and a Fellow of IETE. More information can be found at https://www.bijoycc.site/.

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 Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn. He has been a Professor at Kyoto University, Kyoto, Japan, since 2017. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks. He is a fellow of IEEE and IEICE.