PAPER

CMND: Consistent-Aware Multi-Server Network Design Model for Delay-Sensitive Applications

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

SUMMARY This paper proposes a network design model, considering data consistency for a delay-sensitive distributed processing system. The data consistency is determined by collating the own state and the states of slave servers. If the state is mismatched with other servers, the rollback process is initiated to modify the state to guarantee data consistency. In the proposed model, the selected servers and the master-slave server pairs are determined to minimize the end-to-end delay and the delay for data consistency. We formulate the proposed model as an integer linear programming problem. We evaluate the delay performance and computation time. We evaluate the proposed model in two network models with two, three, and four slave servers. The proposed model reduces the delay for data consistency by up to 31 percent compared to that of a typical model that collates the status of all servers at one master server. The computation time is a few seconds, which is an acceptable time for network design before service launch. These results indicate that the proposed model is effective for delay-sensitive applications.

key words: *data consistency, delay-sensitive service, network design, distributed processing, conservative synchronization, time warp*

1. Introduction

The trend toward lower delay networks is accelerating with the launch of the Fifth-Generation Mobile Communication System (5G) [1] and considering all photonics networks [2]. In addition, with the development of Internet of Things (IoT) services, a variety of applications (APLs) are being provided via a network. Among these services, differences in delay for each user are an issue for space-sharing type APLs, such as network games, in which multiple users share a scenario space. For example, in fighting network games, if the delay for each user varies significantly depending on the distance from an application server, it may significantly influence the games' results.

To address the issue, various solutions are currently being taken according to the characteristics of APLs. For example, in the process of judging hits in shooting games, judging hits at a coordinate position slightly earlier than the current position considering the delay until the player's action (event) arrives at the APL server [3]. The low delay and the guarantee of event order could become major issues in real-time IoT areas such as automated driving and telemedicine.

Distributed processing that considers the ordering of events has been studied in the field of parallel distributed processing. There are two typical algorithms that process events with the guarantee of event order [4]. One is a conservative synchronization, and the other is optimistic synchronization. In the conservative synchronization algorithm, the order of occurrence of events is sequentially guaranteed before application processing by time information to the events. On the other hand, the optimistic synchronization algorithm processes events in order of arrival. If a past event is received, the order of events is guaranteed by the rollback of application status, and the status is modified. Time Warp [5] is a well-known implementation of the rollback process. The causal ordering and the total ordering are generally known as the methods of modifying the order of events. The causal ordering [6] guarantees the order of events, assuming that there is ordering among the received events. On the other hand, the total ordering [7] guarantees that all events on all servers operate in the same order.

The work [8] addressed the guarantee of event order and the delay reduction in distributed processing. The work provides a server selection scheme for distributed processing that minimizes the end-to-end delay and processes events with a guarantee of the occurrence order. The scheme reduces delay compared to the central processing scheme. In the scheme mentioned in [8], each distributed server processes all events of all users in the occurrence order, and it is assumed that each server is always in the same state. However, the possibility cannot be completely ruled out that the order of processing may be changed even if the order arrives at the servers at the same time, depending on link errors, application load conditions, and the state of the input/output (I/O) queue. If only one server processes events in a different order and changes the status differently than the others, the assumption that all servers progress in the same status is broken. If this assumption is broken, users who belong to different servers will not receive the same results and may impede the application's progress. A question grabs our attention: how can we guarantee data consistency among distributed servers with low delay even if a server processes events in a different order?

To address the above question, this paper proposes a consistent-aware multi-server network design model for delay-sensitive applications (CMND). The proposed model

Manuscript received July 3, 2023.
Manuscript revised September 4, 2023.
Manuscript publicized January 30, 2024.

[†]The author is with the Toyohashi University of Technology, Toyohashi-shi, 441-8580 Japan.
[††]The author is with the South Asian University, New Delhi, India.
[†††]The author is with the Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: kawabata.akio.oc@tut.jp

DOI: 10.23919/transcom.2023EBP3112

guarantees that each server maintains the same status, even if a server processes events in a different order. The master server modifies its own status and the statuses of other slave servers to determine the correct status by majority vote. If a mismatch is detected, the master server rollbacks the status to modify the status. In the rollback process, it is necessary to maintain the past status of the application to re-processed events in the occurrence order from the past application status. In CMND, the method for guaranteeing the event order is based on a conservative synchronization algorithm and total ordering. The status modification method is based on the rollback process [5]. The main contributions of CMND are summarized as follows:

- Regardless of the users' location, application processing proceeds to keep the order in which events occur for all users. This means that all users using the application share the same time space.
- Confirm data consistency of processing results on all servers in case an unexpected failure occurs. The rollback process is performed in the case of inconsistency to keep consistency among all servers.
- The sum of the end-to-end delay and the delay for data consistency is minimized by distributing the collating function for data consistency in multiple servers without deciding on one master server.

CMND guarantees data consistency with low delay for delay-sensitive services; it incorporates the time for data consistency as part of the delay and minimizes the total delay. The proposed scheme can be described as a problem of optimizing the allocation of users to servers in order to minimize delay. In order to address this allocation problem, we introduce an integer variable that denotes the selection of servers from the pool of candidate servers. The problem is formulated by employing a binary variable to represent a selection decision for the server; the variable takes a value of 1 to indicate the server's selection and 0 to indicate its non-selection. Hence, the proposed scheme is formulated as an integer linear programming (ILP) problem. As an evaluation of CMND, we evaluate the end-to-end delay of 10, 100, and 500 users under the condition that servers are located at each node in the Kanto area of the Japan Photonic Network Model (JPN) [9] and the ultra-high capacity optical transmission networks (COST) [10]. Numerical results indicate that CMND reduces the delay by 6.8–31.2 percent to maintain data consistency compared to that of a typical model where one master server collects and collates the status of all servers. We also evaluate the computation time of CMND and observe that it is an acceptable time for determining the network design before the service launch. These evaluations indicate that CMND guarantees of event order and data consistency with low delay.

This paper is an extended version of the work [11]. The main additions are as follows. We present related works and the originality of our work. We improve the work [11], which is limited to two slave servers, to allow the number of slave servers specified as a given parameter. By adding this parameter, networks can be designed with flexibility for the conflicting characteristics of high redundancy and low delay. We additionally investigate the evaluation of delay and computation time due to the number of slave servers.

The rest of the paper is organized as follows. Section 2 presents related works and the originality of our work. Section 3 presents the prerequisite and necessity of CMND. In Sect. 4, we formulate CMND as an optimization problem. In Sect. 5, we evaluate CMND in terms of delay and computation time for two types of networks. Finally, Sect. 6 concludes this paper.

2. Related Works

A network design scheme in distributed processing has been studied to reduce the end-to-end delay in applications where multiple users communicate interactively [8]. In the scheme, the user's events are multicast from the server accommodating the users to all other servers, and each distributed server processes all events of all users. At a time called virtual time, each server rearranges and processes all events in the occurrence order so that, in a steady state, all servers maintain the same processing results simultaneously. If a user switches the server with network failures, the same results are maintained on all servers, so transferring user data from the old accommodating server to the new one is unnecessary [12]. On the other hand, it does not consider anomalous situations where data consistency between servers is not maintained, such as when a server accommodating users cannot send events to other servers.

Various methods have been introduced for data replication and consistency in distributed processing systems. Hadoop [13] and Spark [14] are popular for distributed processing of databases, and each has implemented features that take fault tolerance into consideration. In Hadoop, huge amounts of data are located across multiple servers named data nodes. Each data is replicated and located on multiple data nodes designated by a management server named a name node, so that processing can continue even if one data node fails. In Spark, Resilient Distributed Dataset (RDD) manages data replication and consistency; data is recovered even if the data is lost with server failure. It maintains the events that generate the data and re-executes the events to the pre-failure data to recover the data even if the data is lost. This method takes into account data replication and consistent methods for distributed deployment of huge amounts of data that are difficult to process on a single server. It does not target situations where all servers have the same data and run the same applications, as in the proposed model.

In a blockchain, there are characteristics that ensure data replication and consistency by using a distributed processing system and recording the results in a decentralized manner [15]. In terms of data replication and consistency, Raft [16] is the consensus algorithm used in permission-based blockchains composed of trusted nodes. Raft maintains all servers in the same state by applying the same events in order. The events are multicast to all servers. One reader server determines the processing order of each server to obstruct the reversal of the order in arrival events caused by the differences in network delay. This differs from the proposed model, where each server works as a master and collates data in order to minimize the delay.

Compared to these related works, CMND is a new model that minimizes the delay, including the delay of data consistency between multiple servers at the condition of processing the event in occurrence order. In applications where the application space is shared by all users, such as network games and online trading, CMND can realize a network with high tolerance systems without disadvantaging users who are far away from the servers (i.e., those with large delays).

3. Prerequisite and Necessity

3.1 Prerequisite of Distributed Processing and Guarantee of Event Order

We describe the distributed processing and the guarantee of event order assumed in CMND. Each user belongs to one optimal server. Each distributed server multicasts user events to other servers. As shown in Fig. 1, the events of user *a*, who belongs to server 1, are multicasted from server 1 to servers 2 and 3. Thus, each server receives events from all users.

Figure 2 shows an example of the guarantee of event order. Let $D_{\mathrm{U}}^{\max}$ be the maximum delay between user and server and $D_{\mathrm{S}}^{\max}$ be the maximum delay between servers. In the example in Fig. 2, $D_{\mathrm{U}}^{\max}$ is 0.020 [sec], delay between user *a* and server 1, and $D_{\mathrm{S}}^{\max}$ is 0.005 [sec], delay between server 1 and server 3. All events are rearranged with $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$ delay from the time of occurrence at each server. In server 3, the event of user *a* arrives with a delay of 0.025 [sec] (=0.020 [sec]+0.005 [sec]), so there is no waiting. Event of user *b* arrives at server 3 with a delay of 0.015 [sec] (=0.010 [sec]+0.005 [sec]), so waits for 0.010 [sec] (=0.025 [sec]-0.015 [sec]). Event of user *c* is processed similarly, resulting in all events being rearranged at $T + D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$. These processes are performed for all events at each server.

To guarantee that users also receive the results at the same time, the arrival time of all users is controlled by each server to arrive at the user after a maximum delay between the user and server of $D_{\mathrm{U}}^{\max}$. Thus, all users receive the results processed at $T+D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}$ with a delay of $D_{\mathrm{U}}^{\max}$. Therefore, the end-to-end delay of each user is $T + 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$.

3.2 Necessity of Data Consistency

We describe the necessity of data consistency. In the prerequisite distributed processing, all servers receive the same events in the same order at applications, as described in Sect. 3.1. From these conditions, the status coincides across all servers, and each user receives the same results even if they belong to different servers. However, the possibility cannot be completely ruled out that the order of processing

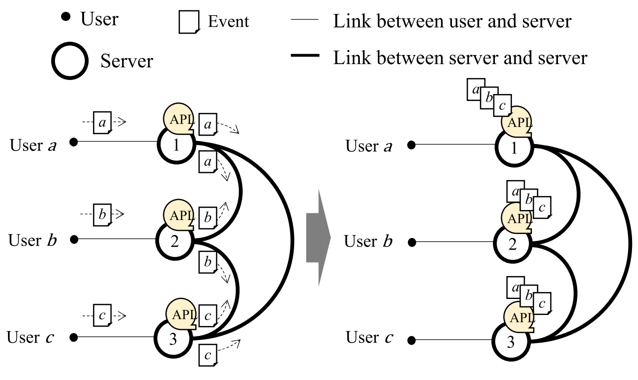


Fig. 1 Example of prerequisite distributed processing.

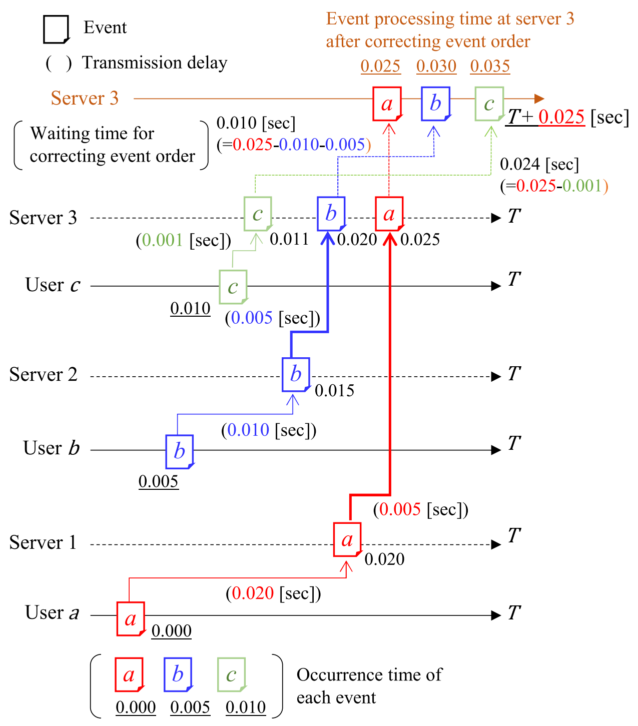


Fig. 2 Example of guarantee of event order at server 3.

may change even if the order of arrival at the servers is the same. Especially, data consistency is necessary for APLs such as network games and online trading, where the status in data consistency at each server would alienate the progress of the scenario.

When the status among servers is mismatched, it is necessary to modify the status to guarantee data consistency. The issues are how to determine the correct status and how to modify it. To determine the correct status, more statuses are desirable to collate since it is uncertain which status is wrong in a 1:1 collation. The status modification method uses the rollback process that rewinds the APL status before the events are received and reprocesses the event to modify it to the correct status [17], [18]. From the user's perspective, rollback impacts the application's quality. Since the past status is modified, the rewinding time and number of rollbacks should be reduced for application quality.

4. Proposed Model

4.1 Overview

In this section, we describe CMND. The correct status is determined by a majority vote collating the status of three or more servers. Each server collates its own status and the status of other slave servers, with itself as the master server. The master server selects the slave servers to minimize the time for data consistency. If the status does not match between the collating servers, the rollback process is initiated and the status is modified. Figure 3 shows an example of this process. Server 3 is the master server and servers 1 and 2 are slave servers. In server 3, the status is rollbacked to the time the events are received and modified to the correct status.

We describe the delay of CMND. The time for data consistency of the master server is delayed by the maximum delay between the master server and the slave servers. L_i denotes the delay for data consistency in server i. $L^{\max}$ is the maximum value of L_i over all servers $i \in V_{\mathrm{S}}$, i.e., $L^{\max} = \max_{i \in V_{\mathrm{S}}} L_i$, where V_{S} is the set of servers. Figure 4(b) shows the master and slave server pairs when the selected servers are as shown in Fig. 4(a). In Fig. 4, L_i of servers 1, 2, 3, and 4 are 5, 3, 3, and 5, respectively, and $L^{\max}$ is 5. Since all servers are processing events at $T + D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$, the delay considering data consistency is $D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + L^{\max}$. To guarantee that all users receive the results from the accommodating server with the same delay, the timing of sending the results is adjusted so that all results arrive at the users with a delay of $D_{\mathrm{U}}^{\max}$. Therefore, all users are using the application at $T + 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + L^{\max}$. In CMND, the selected servers and the master-slave server pairs are determined to minimize $2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + L^{\max}$. We introduce a parameter, N_{SS}, that is the number of slave servers. If N_{ss} is increased for redundancy and the number of slave servers is increased, $L^{\max}$ tends to increase as well. For example, when $N_{\mathrm{ss}} = 1$, one slave server is required, but when $N_{\mathrm{ss}} = 2$, two slave servers are required, and the second one is more likely that a server is farther away than the first one.

4.2 Formulation

We formulate CMND as an ILP problem. The network is represented as an undirected graph $G(V, E)$. Let V be the set of nodes as users and servers, and E be the set of undirected links. $V_{\mathrm{U}} \subseteq V$ is the set of users and $V_{\mathrm{S}} \subseteq V$ is the set of servers, where $V_{\mathrm{U}} \cup V_{\mathrm{S}} = V$ and $V_{\mathrm{U}} \cap V_{\mathrm{S}} = \emptyset$. $E_{\mathrm{U}} \subseteq E$ is the set of links between user and server, and a link between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$ is denoted by $(p, i) \in E_{\mathrm{U}}$. $E_{\mathrm{S}} \subseteq E$ is the set of links between server and server, and a link between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}}$ is denoted by $(i, j) \in E_{\mathrm{S}}$, where $E_{\mathrm{U}} \cup E_{\mathrm{S}} = E$ and $E_{\mathrm{U}} \cap E_{\mathrm{S}} = \emptyset$. It is assumed that $(p, i) \in E_{\mathrm{U}}$ is set between every user $p \in V_{\mathrm{U}}$ and every server $i \in V_{\mathrm{S}}$. $(i, j) \in E_{\mathrm{S}}$ is assumed to be set between all servers $i \in V_{\mathrm{S}}$ and all servers $j \in V_{\mathrm{S}}$ (but $i \neq j$).

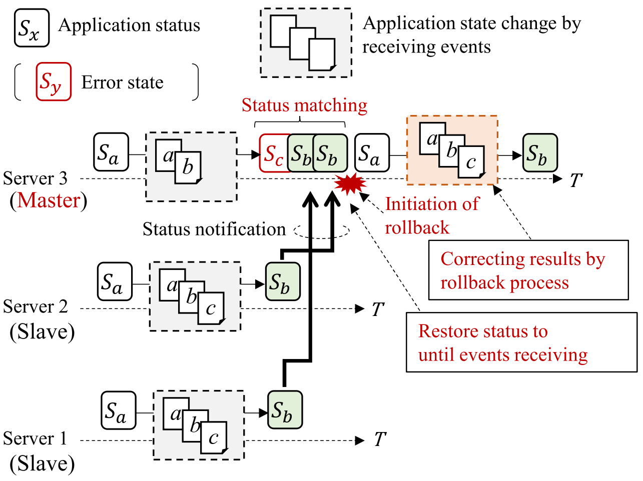


Fig. 3 Example of rollback process.

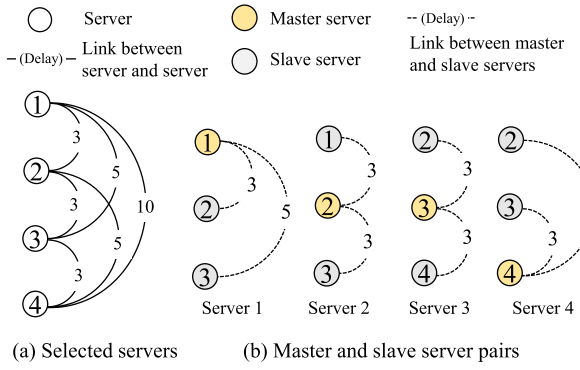


Fig. 4 Example of master and slave server pairs for CMND.

Let $d_{pi}, (p, i) \in E_{\mathrm{U}}$, be the delay between user $p \in V_{\mathrm{U}}$ and server $i \in V_{\mathrm{S}}$. Let $d_{ij}, (i, j) \in E_{\mathrm{S}}$, be the delay between server $i \in V_{\mathrm{S}}$ and server $j \in V_{\mathrm{S}}$. It is assumed that the user belongs to one server, and $M_i, i \in V_{\mathrm{S}}$, is the maximum number of users that server i can be accommodated. $x_{kl}, (k, l) \in E$, is a binary variable; $x_{kl} = 1$ if link $(k, l) \in E$ is selected, and $x_{kl} = 0$ otherwise. $y_i, i \in V_{\mathrm{S}}$, is a binary variable; $y_i = 1$ if the server i is selected by at least one user, and $y_i = 0$ otherwise. $z_{ij}, (i, j) \in E_{\mathrm{S}}$, is a binary variable; $z_{ij} = 1$ if the link (i, j) is selected as a link between the master and slave servers, and $z_{ij} = 0$ otherwise. $L^{\max}$ is the maximum delay between each server and the corresponding slave servers. That is a delay in data consistency. N_{SS} is the number of slave servers that will make the major vote to guarantee data consistency.

We formulate the network topology decision problem for CMND as an ILP problem, which is given by:

$$\text{Objective min} \quad 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + L^{\max} \tag{1a}$$

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

$$\sum_{p \in V_{\mathrm{U}}} x_{pi} \leq M_i, \forall i \in V_{\mathrm{S}} \tag{1c}$$

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

Table 1 Given parameters and decision variables.

Given parameters	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
	M_i	Maximum number of users for server $i \in V_{\mathrm{S}}$ can be accommodated
	N_{SS}	Number of slave servers
	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 $j \in V_{\mathrm{S}}$
Decision variables	$D_{\mathrm{U}}^{\max}$	Maximum delay between user and server
	$D_{\mathrm{S}}^{\max}$	Maximum delay between server and server
	$L^{\max}$	Delay for data consistency
	x_{pi}	$x_{pi} = 1$ if link $(p, i) \in E_{\mathrm{U}}$ is selected and $x_{pi} = 0$ otherwise
	y_i	$y_i = 1$ if server $i \in V_{\mathrm{S}}$ is selected and $y_i = 0$ otherwise
	z_{ij}	$z_{ij} = 1$ if link $(i, j) \in E_{\mathrm{S}}$ is used for master-slave link and $z_{ij} = 0$ otherwise

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

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

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

$$d_{pi} x_{pi} \leq D_{\mathrm{U}}^{\max}, \forall (p,i) \in E_{\mathrm{U}} \tag{1h}$$

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

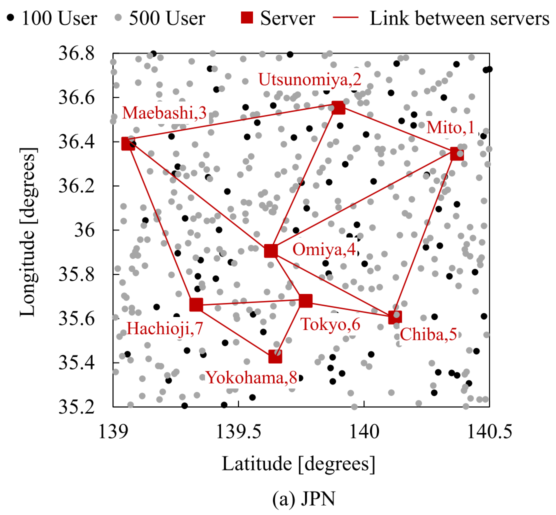
$$\sum_{i \in V_{\mathrm{S}}} y_i \geq N_{\mathrm{SS}} + 1 \tag{1j}$$

$$z_{ij} \leq x_{ij}, \forall (i,j) \in E_{\mathrm{S}} \tag{1k}$$

$$d_{ij} z_{ij} \leq L^{\max}, \quad \forall i \in V_{\mathrm{S}}, j : (i,j) \in E_{\mathrm{S}} \tag{1l}$$

$$\sum_{j:(i,j) \in E_{\mathrm{S}}} z_{ij} = y_i N_{\mathrm{SS}}, \forall i \in V_{\mathrm{S}}. \tag{1m}$$

Given parameters and decision variables of CMND are shown in Table 1. Equation (1a) indicates that the delay $2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + L^{\max}$ is minimized as the objective function. Equation (1b) indicates that the user selects optimal one user-server link from the candidate links between the user and all servers. Equation (1c) indicates that the number of users accommodated by server $i \in V_{\mathrm{S}}$ does not exceed M_i. Equation (1d) indicates that if $x_{ij} = 1$ and link $(i, j) \in E_{\mathrm{S}}$ is selected, server $i \in V_{\mathrm{S}}$ is also selected and $y_i = 1$: a server with at least one selected link is the selected server. Equations (1e)–(1g) are linear expressions, which indicate $y_i y_j = x_{ij}$: a link between the selected servers is the selected link. Equation (1h) indicates that the maximum value of d_{pi} of the selected link is $D_{\mathrm{U}}^{\max}$. Equation (1i) indicates that the maximum value of d_{ij} of the selected link is $D_{\mathrm{S}}^{\max}$. Equation (1j) indicates that there are at least $N_{\mathrm{SS}} + 1$ servers to be selected for majority voting. Equation (1k) indicates that the link used as the link between the master and slave server is selected from the links satisfying $x_{ij} = 1$. Equation (1l) indicates that the maximum value of delay for each selected server and the slave servers is $L^{\max}$. Equation (1m) indicates that each selected server has N_{SS} slave servers.



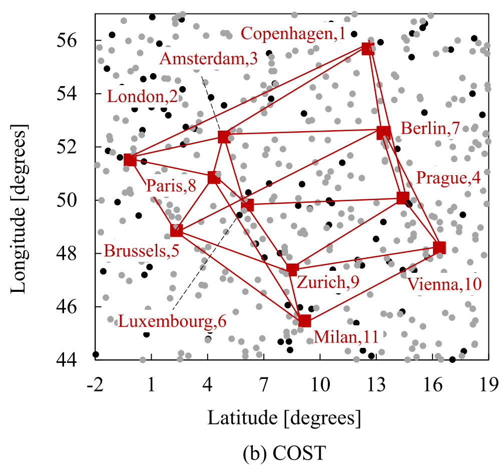


Fig. 5 Location of users and servers for JPN and COST.

5. Numerical Evaluation and Discussion

In this section, we describe the evaluation of CMND. We compare the delay of CMND with that of a typical master server model as a benchmark. In the master server model, one master server is selected from the candidate master servers and collates the status of all servers. The delay of the master server model is obtained using the ILP problem of (A· 1a)–(A· 1g) in Appendix A. The network assumes that the servers are located at the Kanto area of JPN [9] and the COST [10] nodes. 10, 100, and 500 users are uniformly distributed in the square region. Figures 5(a) and (b) show the location of users and servers for JPN and COST, respectively. The delay is assumed to be proportional to transmission distance. In addition to transmission delays, actual delays include queuing delays at switches and processing delays at servers. Since these delays are influenced by the number of facility resources, service providers usually provide sufficient processing power and bandwidth for delay-sensitive applications. In this evaluation, we focus on the network topology design of users and servers and consider only transmission delays, assuming that a sufficient amount of facilities are provided. The distances of the user-server link and the server-server link are calculated from the linear distance based on latitude and longitude, and 100 km is assumed as a delay of 0.5 [ms]. It is assumed that each server is connected by a full mesh with the shortest path. For example, in Fig. 5(a), the Yokohama node is connected to the Chiba node via the Tokyo node. Our evaluation uses CPLEX [19] on an Intel(R) Xeon(R) Gold 6132 CPU 2.60 GHz, 32 GB memory.

Figure 6 shows the delay in the JPN and the COST for CMND and the master server model. The delay without data consistency is also shown for reference. The delay without data consistency is obtained using the ILP problem of (A· 2a)–(A· 2b) in Appendix B. The number of slave servers is evaluated with two, three, and four slave servers. In the master server model, as a benchmark, one master server is selected from the candidate master servers to minimize the delay. To compare the delay of CMND with the lowest delay of the master server model, we evaluate each delay expressed in Eq. (A· 1b) when each server is set to the master server. In CMND, the delay increases as the number of slave

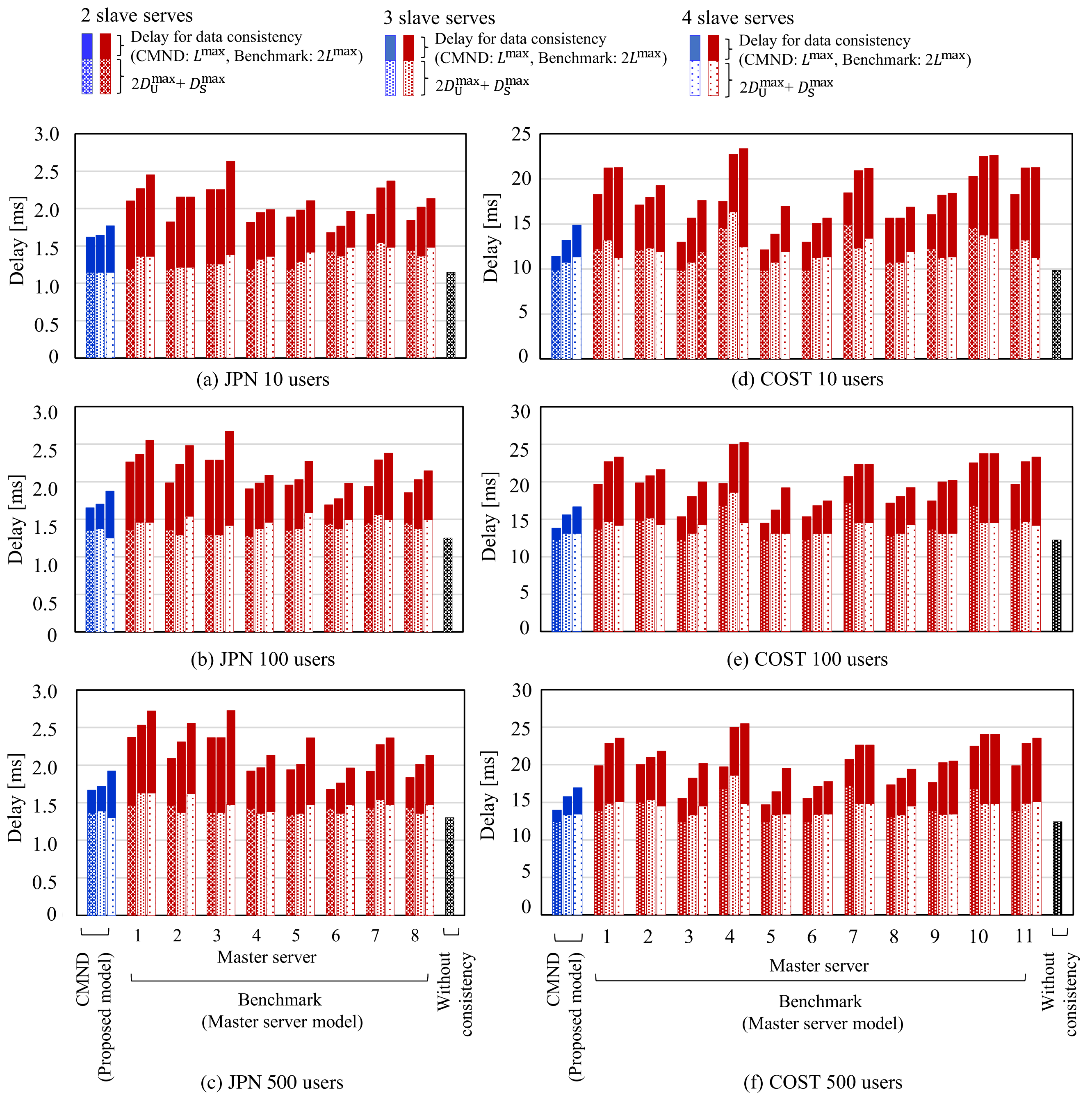


Fig. 6 Delay of CMND and benchmark for JPN and COST.

servers increases. This is because the maximum delay for data consistency increases as the number of slave servers increases. In the master server model, where one master server collects and collates the application status with that of other slave servers, the delay may increase as the number of slave servers increases. This is because the delay may increase as the number of selected servers increases by the constraint of Eq. (1j) in Appendix A. For 10 users in JPN, the additional delay of CMND from the delay without data consistency is 0.471, 0.499, and 0.625 [ms] in two, three, and four slave servers, respectively. The smallest value for that of the master server model is 0.537, 0.623, and 0.824 [ms] in two, three, and four slave servers, respectively. Thus, the additional delay of CMND is 88, 80, and 76 percent compared to that of the master server model in two, three, and four slave servers, respectively. For 100 users in JPN, the additional delay of CMND is 90, 86, and 86 percent compared to that of the master server model in two, three, and four slave servers, respectively. For 500 users in JPN, the additional delay of CMND is 98, 91, and 95 percent compared to that of the master server model in two, three, and four slave servers, respectively. For 10, 100, and 500 users in COST, the additional delay of CMND is up to 96 percent (three slave servers for 100 users) and down to 69 percent (two slave servers for 10, 100, and 500 users) compared to the delay of the master server model. The delay may increase with an increase in the number of users. This increase in delay is due to the possibility of more users with large d_{pi} rather than the number of users since the user with the largest delay determines $D_{\mathrm{U}}^{\mathrm{MAX}}$.

From these results, CMND reduces the additional delay compared to that of the master server model in all scenarios. In CMND, each selected server, as a master server, collects the status and modifies its own status if a mismatch is found. In other words, the delay for data consistency is the maximum delay in one direction between each server and the corresponding slave servers. In the master server method, one master server collects all server statuses and sends the modification order to the mismatched server if a mismatch is found. In other words, the delay for data consistency is a maximum round-trip delay between the master server and the other servers. In this way, the distribution for the status collation process and the status modification process reduces the delay for data consistency in CMND. CMND is effective in reducing delay regardless of the network topology and the number of slave servers, and it is effective in data consistency with low delay.

We discuss the use cases. In the scenario where a single network operator facilitates the networks of multiple application providers, it is plausible for the network operator to offer servers within its own cloud infrastructure. The network operator offers an optimized server and network configuration designated by the service provider, while the application providers utilize the servers that have been provided to them. In the scenario where an application provider utilizes its own servers, it determines the most suitable configuration that the network operator provides and its own multiple servers.

Table 2 Computation times.

Users	No. of slave servers	Ave. time (min-max) [sec]
JPN 500 users	2	1.56 (1.55–1.58)
	3	2.20 (2.14-2.27)
	4	1.40 (1.31-1.47)
COST 500 users	2	4.06 (3.94–4.15)
	3	7.87 (7.58-8.06)
	4	4.31 (4.23-4.39)

We describe the computation time of CMND. Table 2 shows the average computation times for five trials of 500 users in the JPN and the COST, which is an acceptable time for network design before service launch. If participants are predetermined, it allows ample time to plan and design the network before providing services. In scenarios where participants are predetermined prior to the commencement of service, such as in the context of participatory online games, a waiting period of approximately 10 seconds is widely regarded as tolerable in numerous instances.

6. Conclusion

This paper proposed a network design model considering data consistency for a delay-sensitive distributed processing system. The model achieves the guarantee of event order and data consistency with low delay. In the proposed model, each distributed server selects slave servers for collating whether the status is correct or not. If the status is incorrect, the rollback process is initiated to modify the application status to guarantee data consistency. The select servers and the master-slave server pairs are determined to minimize the end-to-end delay and delay for data consistency. We formulated the proposed model as an ILP problem and evaluated the delay performance at the condition that the servers are located in the Kanto area of JPN and the COST node. We compared the delay of the proposed model with that of a typical master server model as a benchmark. The additional delay of the proposed model is reduced by up to 24 percent for JPN and 31 percent for COST compared to the master server model. In addition, the computation time was a few seconds, which is an acceptable time for network design before service launch. The proposed model aims to design a distributed processing network in order to guarantee the event order with low delay and data consistency. For applications for space-sharing type APLs, such as network games and online trading, these results indicate that the proposed model is effective in guaranteeing data consistency with low delay.

Acknowledgments

This work was supported by JSPS KAKENHI JP23K16867. 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 26 June 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. DOI: 10.53829/ntr202110fa1

[3] Y.W. Bernier, "Latency compensating methods in client/server in-game protocol design and optimization," Game Developers Conference, vol.98033, no.425, 2001.

[4] R.M. Fujimoto, Parallel and Distributed Simulation Systems, John Wiley, New York, 2000.

[5] D.R. Jefferson, "Virtual time," ACM Transactions on Programming Languages and Systems (TOPLAS), vol.7, no.3, pp.404–425, 1985. DOI: 10.1145/3916.3988

[6] K. Birman, A. Schiper, and P. Stephenson, "Lightweight causal and atomic group multicast," ACM Transactions on Computer Systems (TOCS), vol.9, no.3, pp.272–314, 1991. DOI: 10.1145/128738.128742

[7] R. Baldoni, S. Cimmino, and C. Marchetti, "A classification of total order specifications and its application to fixed sequencer-based implementations," Journal of Parallel and Distributed Computing, vol.66, no.1, pp.108–127, 2006. DOI: 10.1016/j.jpdc.2005.06.021

[8] 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. DOI: 10.1109/ACCESS.2017.2758803

[9] "Japan Photonic Network Model," https://www.ieice.org/cs/pn/eng/JPNM/TokyoMANModel.zip, online; online; accessed 26 June 2023.

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

[11] A. Kawabata, B.C. Chatterjee, S. Ba, and E. Oki, "A network design approach considering data consistency for delay-sensitive distributed processing systems," IEEE International Conference on Communications (ICC), pp.3060–3065, 2023.

[12] A. Kawabata, B.C. Chatterjee, and E. Oki, "MHND: Multi-homing network design model for delay sensitive applications," IEICE Trans. Commun., vol.E106-B, no.11, pp.1143–1153, Nov. 2023.

[13] T. White, Hadoop: The Definitive Guide, O'Reilly Media, 2012.

[14] "Apache Spark," https://spark.apache.org/, online; accessed 26 June 2023.

[15] "Blockchain," https://www.blockchain.com/, online; accessed 26 June 2023.

[16] M. Mazzoni, A. Corradi, and D.V. Nicola, "Performance evaluation of permissioned blockchains for financial applications: The ConsenSys Quorum case study," Blockchain: Research and Applications, vol.2, no.1, p.100026, 2022. DOI: 10.1016/j.bcra.2021.100026

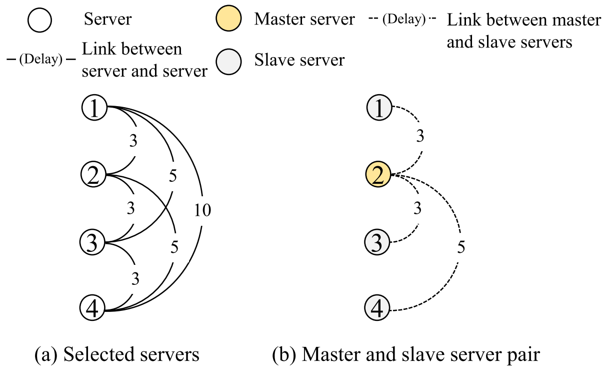
[17] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-lag and timewarp: Providing consistency for replicated continuous applications," IEEE Trans. Multimedia, vol.6, no.1, pp.47–57, 2004. DOI: 10.1109/TMM.2003.819751

[18] A. Kawabata, B.C. Chatterjee, and E. Oki, "An optimistic synchronization based optimal server selection scheme for delay sensitive communication services," IEICE Trans. Commun., vol.E104-B, no.10, pp.1277–1287, Oct. 2021. DOI: 10.1587/transcom.2020EBP3178

[19] "IBM ILOG CPLEX," https://www.ibm.com/analytics/cplex-optimizer, online; accessed 26 June 2023.

Appendix A: Formulation of Master Server Model

In the master server model, one master server collates the status of all servers. The slave server sends the processing results to the master server. The master server is selected to minimize the delay with all servers. Figure A· 1(b) shows an example of the master server and slave servers when the server in Fig. A· 1(a) is selected. In the example in Fig. A· 1, server 2 is selected as the master server to minimize the delay with all servers, and the maximum delay of the link between the master and slave servers is $L^{\max} = 5$.



Server
Master server
Link between master and slave servers
(Delay)
Link between server and server
Slave server
1
2
3
4
3
5
10
3
5
3
(a) Selected servers
(b) Master and slave server pair

Fig. A· 1 Example of master and slave server pair for master server model.

The following given parameter and decision variable are added to those in Sect. 4.2. Since the master server collates the status of all servers, it is required to have sufficient performance. We specify the candidate servers that can be the master server. The selected servers for the distributed processing and one master server are determined to minimize the delay. We express the candidate servers for the master server with the parameter of $s_i, i \in V_\mathrm{S}$; server i is a candidate of the master server if $s_i = 1$, and server i is not a candidate of the master server if $s_i = 0$. $m_i, i \in V_\mathrm{S}$, is a binary variable; $m_i = 1$ if server i is selected as a master server, and $m_i = 0$ otherwise.

The server selection problem for the master server model is formulated as an ILP problem as follows:

$$\text{Objective} \quad \min \quad 2D_\mathrm{U}^{\max} + D_\mathrm{S}^{\max} + 2L^{\max} \tag{A· 1a}$$

$$\text{s.t.} \quad m_i \leq y_i, \forall i \in V_\mathrm{S} \tag{A· 1b}$$

$$m_i + y_j - 1 \leq z_{ij}, \forall (i,j) \in E_\mathrm{S} \tag{A· 1c}$$

$$z_{ij} \leq m_i, \forall (i,j) \in E_\mathrm{S} \tag{A· 1d}$$

$$z_{ij} \leq y_j, \forall (i,j) \in E_\mathrm{S} \tag{A· 1e}$$

$$\sum_{i \in V_\mathrm{S}} s_i m_i = 1 \tag{A· 1f}$$

$$(1a) - (1l). \tag{A· 1g}$$

Equation (A· 1a) shows that the delay $2D_\mathrm{U}^{\max} + D_\mathrm{S}^{\max} + 2L^{\max}$ is minimized as the objective function. The processing results of all slave serves are sent to the master server. If the master server finds a status mismatch, the master server sends a modification order to the mismatched server. Therefore the maximum delay for data consistency is $2L^{\max}$ since the delay for collating and modifying the status is the round trip time between the master server and the mismatched server. Equation (A· 1b) indicates that the master server is selected from the servers to be selected. Equations (A· 1c)–(A· 1e) are linear expressions showing that $m_i y_j = z_{ij}$. Equation (A· 1f) indicates that the number of the master server is one.

Appendix B: Formulation of Server Selection without Data Consistency

An ILP problem as the server selection problem without data consistency is given by:

$$\text{Objective } \min \quad 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} \tag{A· 2a}$$

$$(1a) - (1j). \tag{A· 2b}$$

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 is 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.

Bijoy Chand Chatterjee is an Assistant Professor and DST Inspire Faculty 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 is 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 an IEEE Fellow.