A Network Design Approach Considering Data Consistency for Delay-Sensitive Distributed Processing Systems

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

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

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

***Abstract*—This paper proposes a network design approach considering data consistency for a delay-sensitive distributed processing system. The data consistency is determined by collating the own state and the two 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 approach, the select 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 approach as an integer linear programming problem. We evaluate the delay performance and computation time. The proposed approach reduces the delay for data consistency by 6.8-31.2% compared to that of a typical approach that collates the status of all servers at the 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 approach is effective for delay-sensitive applications.**

***Index Terms*—Data consistency, delay-sensitive service, network design, distributed processing, conservative synchronization**

I. Introduction

The trend toward lower delay networks is accelerating with the launch of 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 takes into account 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 correcting 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] is a study that addressed the guarantee of event order and the delay reduction in distributed processing. This work provided a server selection scheme for distributed processing to minimize the end-to-end delay with the guarantee of event 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, for the first time, proposes a network design approach considering data consistency for delay-sensitive distributed processing systems. The proposed approach 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. For the rollback process, it is necessary to keep the past status of the application. In the proposed approach, the method for the guarantee of event order is based on a conservative synchronization algorithm and total ordering. The method for modifying the status is based on the rollback process [5].

The proposed approach guarantees data consistency with low delay for delay-sensitive services; it incorporates the time for data consistency as part of the delay. We formulate the proposed approach as an integer linear programming (ILP) problem. As an evaluation of the proposed approach, 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 Japan Photonic Network Model 48 (JPN) [9] and ultra-high capacity optical transmission networks (COST) [10]. Numerical results indicate that the proposed approach reduces the delay by 6.8-31.2% to maintain data consistency compared to that of a typical approach that collates the status of all servers at the master server. We also evaluate the computation time of the proposed approach and observe that it is an acceptable time for determining the network design before the service launch. These evaluations indicate that the proposed approach achieves the guarantee of event order and data consistency with low delay.

The rest of the paper is organized as follows. Section II presents the prerequisite and necessity of the proposed approach. In Section III, we formulate the proposed approach as an optimization problem. In Section IV, we evaluate the proposed approach in terms of delay and computation time for two types of networks. Finally, Section V concludes this paper.

II. Prerequisite and Necessity

A. *Prerequisite of distributed processing and guarantee of event order*

We describe the distributed processing and the guarantee of event order assumed in the proposed approach. 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 25 [min], delay between user a and server 3, and $D_{\mathrm{S}}^{\max}$ is 5 [min]. 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 25 [min] (=20 [min]+5 [min]), so there is no waiting. Event of user b arrives at server 3 with a delay of 15 [min] (=10 [min]+5 [min]), so waits for 10 [min] (=25 [min]-15 [min]). Event of user c is processed similarly, resulting in all events

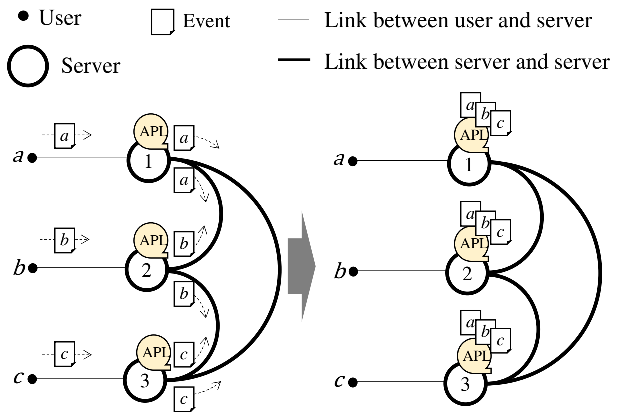


Fig. 1. Example of prerequisite distributed processing.

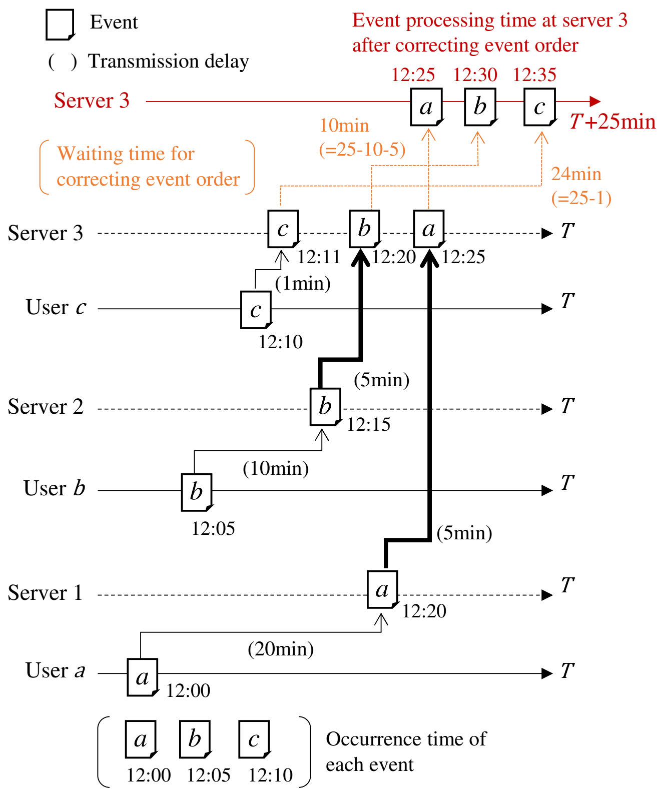


Fig. 2. Example of guarantee of event order at server 3.

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 $T+2D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}$.

B. *Necessity of data consistency*

We describe the necessity of data consistency. In the prerequisite distributed processing, all servers receive the same

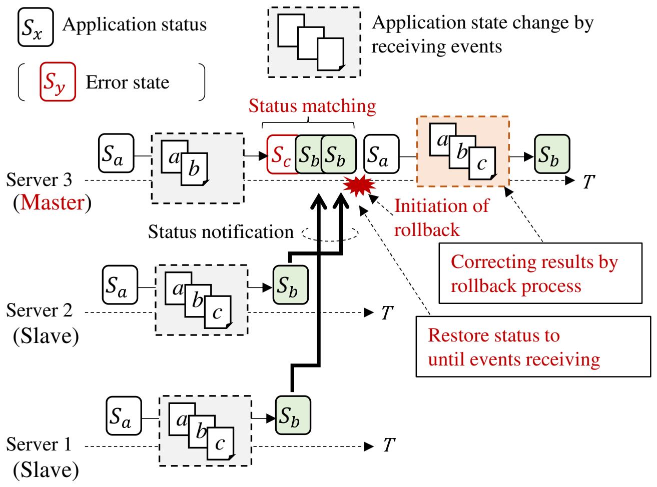


Fig. 3. Example of rollback process.

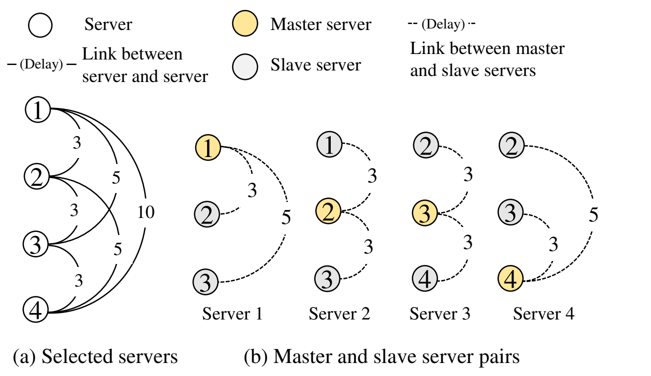


Fig. 4. Example of master and slave server pairs for proposed approach.

events in the same order at applications, as described in Section II-A. 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 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 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 correct 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 method of status modification uses a rollback process that rewinds the APL status to before the event was received and reprocesses the event [11], [12]. 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.

III. Proposed approach

A. Overview

In this section, we describe the proposed approach. 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. To minimize the time for data consistency, the master server selects the two slave servers, the nearest server, and the second nearest server. If the status is mismatched with that of other 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. Server 3 rollback the status to before the events were received and modified it to the correct status.

We describe the delay of the proposed approach. Since each server selects two slave servers, the nearest server and the second nearest server, the data consistency time of the master server is delayed by the transmission delay between the master server and the second nearest server. If the delay between the server i and the second nearest server is L_i, the maximum delay for data consistency among all selected servers is denoted by $L^{\max}$. Figure 4(b) shows the master and slave servers 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 users receive the results at the same time, the timing of sending the results is adjusted so that all results arrive to 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 the proposed approach, the selected servers and the master-slave server pairs are determined to minimize $2D_{\mathrm{U}}^{\max}+D_{\mathrm{S}}^{\max}+L^{\max}$.

B. Formulation

We formulate the proposed approach 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. We denote a user as $p\in V_{\mathrm{U}}$ and $V_{\mathrm{U}}\subseteq V$. We denote a server as $i\in V_{\mathrm{S}}$ and $V_{\mathrm{S}}\subseteq V$. Since there are only users and servers, $V_{\mathrm{U}}\cup V_{\mathrm{S}}=V$. Since there can be no nodes that are users and servers, $V_{\mathrm{U}}\cap V_{\mathrm{S}}=\emptyset$. The link between user $p\in V_{\mathrm{U}}$ and server $i\in V_{\mathrm{S}}$ is denoted by $(p,i)\in E_{\mathrm{U}}$, and $E_{\mathrm{U}}\subseteq E$ is the set of links (p,i). $E_{\mathrm{U}}\subseteq E$ 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 $(p,i)\in E_{\mathrm{U}}$. It is assumed that $(p,i)\in E_{\mathrm{U}}$ is set between every user $p\in V_{\mathrm{U}}$ and every server $i\in V_{\mathrm{S}}$. $E_{\mathrm{S}}\subseteq E$ 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 $(i,j)\in E_{\mathrm{S}}$. $(i,j)\in E_{\mathrm{S}}$ is assumed to be set between all servers $i\in V_{\mathrm{S}}$ and all servers $j\in V_{\mathrm{S}}$ (but $i\neq j$). Since there are only users and servers, $E_{\mathrm{U}}\cup E_{\mathrm{S}}=E$, and since there can be no nodes that are users and servers, $E_{\mathrm{U}}\cap E_{\mathrm{S}}=\emptyset$.

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, $x_{kl} = 0$ 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, $y_i = 0$ 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, $z_{ij} = 0$ otherwise. $L^{\max}$ is the maximum delay with the second nearest server among the selected servers. That is a delay in data consistency.

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

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

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

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

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

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

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

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

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

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

$$\sum_{i \in V_{\mathrm{S}}} y_i \geq 3 \tag{1j}$$

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

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

$$\sum_{j:(i,j) \in E_{\mathrm{S}}} z_{ij} = 2y_i, \forall i \in V_{\mathrm{S}}. \tag{1m}$$

TABLE I
GIVEN PARAMETERS AND DECISION VARIABLES.

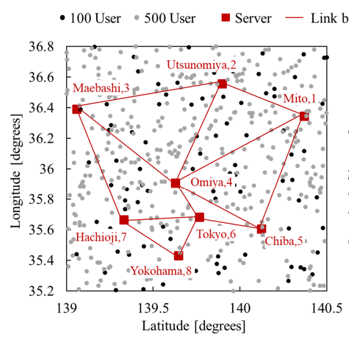
Given parameters	M_i	Maximum number of users of server i can be accommodated
	d_{pi}	Delay between user p and server i.
	d_{ij}	Delay between server i and j
Decision variables	$D_{\mathrm{U}}^{\max}$	Maximum delay between user and server
	$D_{\mathrm{S}}^{\max}$	Maximum delay between server and server
	$L^{\max}$	Delay for data consistency
	x_{pi}	Link (p,i) is selected or otherwise
	y_i	Server i is selected or otherwise
	z_{ij}	Link (i,j) is used for master-slave link or otherwise

Given parameters and decision variables of the proposed approach are shown in Table I. 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 three 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 the second nearest server among the selected servers is $L^{\max}$. Equation (1m) indicates that the selected server has two slave servers.

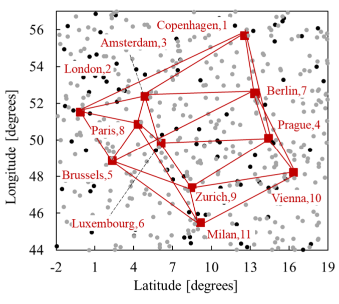
IV. Numerical evaluation and discussion

In this section, we describe the evaluation of the proposed approach. We compare the delay of the proposed approach with that of a typical master server approach as a benchmark. In the master server approach, a master server is determined among selected servers and collates the status of all servers at the master server. The delay of the master server approach is obtained using the ILP problem of (2a)-(2g) in Appendix A. The network assumes that the servers are located at the JPN and COST 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 [13] on an Intel(R) Core(TM) i7-9750H CPU 2.60 GHz, 16 GB memory.

Figure 6 shows the delay in the JPN for the proposed approach and the master server approach. The delay without data consistency is also shown for reference. The delay without data consistency is obtained using the ILP problem of (3a)-(3b) in Appendix B. For 10 users, the additional delay from the delay without data consistency is 0.471 [ms] for the proposed approach, and the smallest value for that of the master server



(a) JPN



(b) COST

Fig. 5. Location of User and server for JPN and COST.

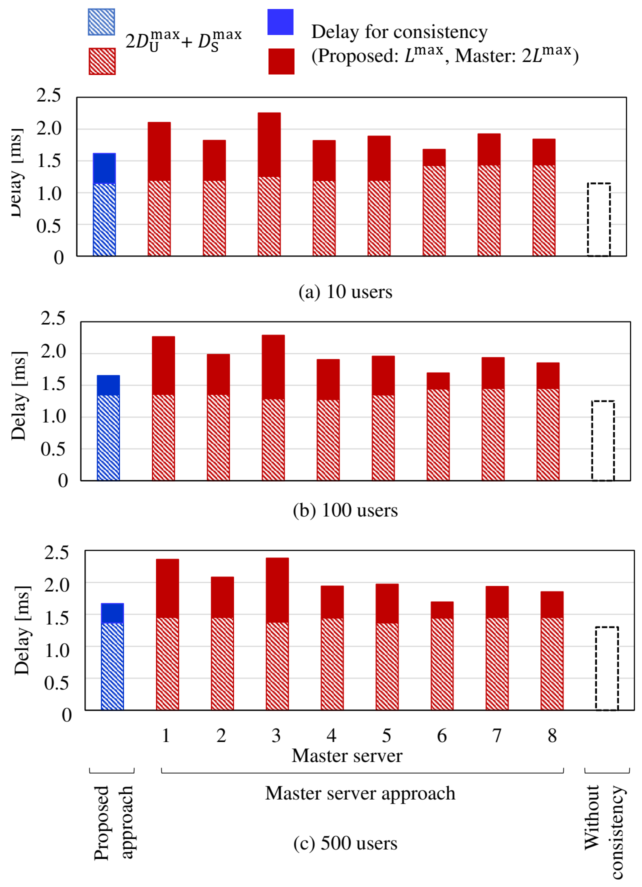


Fig. 6. Delay of proposed approach and master server approach for JPN.

approach is 0.537 [ms] (master server is server 6). For 100 users, the additional delays are 0.403 [ms] for the proposed approach and 0.444 [ms] (master server is server 6) for the master server approach. For 500 users, the additional delays are 0.369 [ms] (master server is server 6) for the proposed approach and 0.396 [ms] for the master server approach. The additional delays of the proposed approach are reduced by 6.8-12.3% compared to the smallest value of that of the master server approach. Figure 7 shows the delay for 10, 100, and 500 users in the COST. The additional delays of

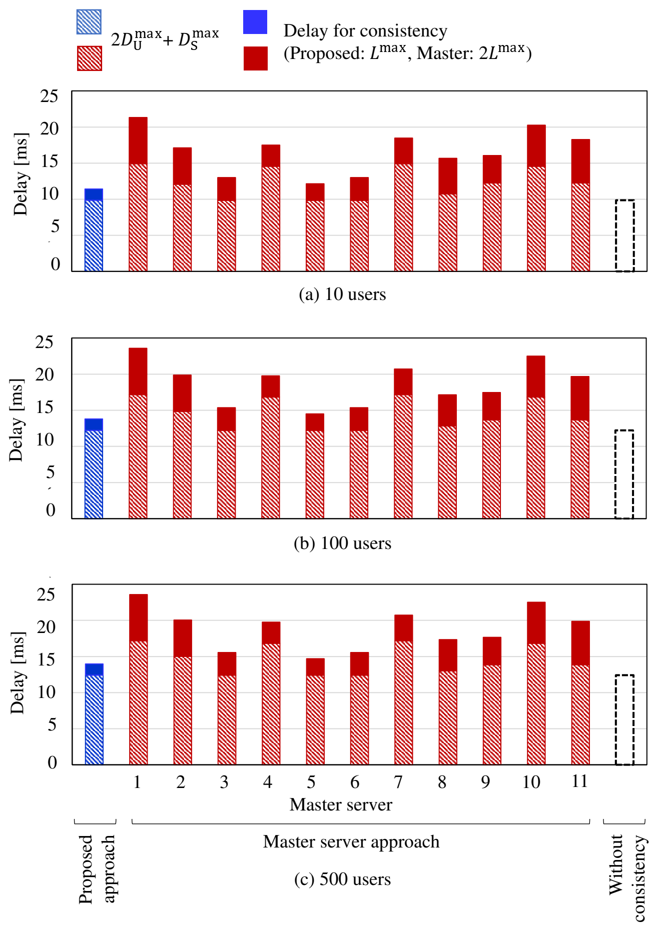


Fig. 7. Delay of proposed approach and master server approach for COST.

the proposed approach are reduced by 31.2% for 10, 100, and 500 users, compared to that of the smallest value (master server is server 5) of the master server approach.

From these results, the proposed approach reduces the additional delay compared to that of the master server approach in all scenarios. The proposed approach is effective in reducing delay regardless of the network topology, and it is effective in data consistency with low delay.

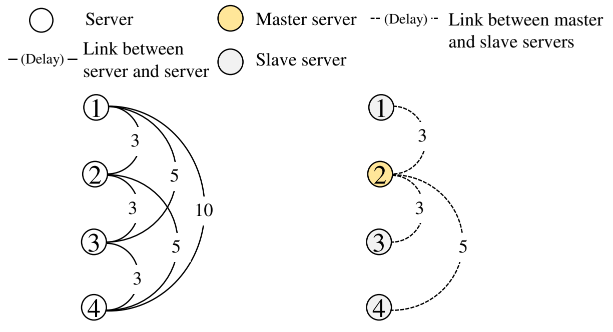
We describe the computation time of the proposed approach: the average computation times for five runs of 500 users in the JPN and COST are 1.07 [sec] and 2.42 [sec], respectively, which is an acceptable time for network design before service launch.

V. Conclusion

In this paper, we proposed a network design approach considering data consistency for a delay-sensitive distributed processing system. The approach achieves the guarantee of event order and data consistency with low delay. In the proposed approach, each distributed server selects two slave servers for collating whether the status is correct or not. If the status is incorrect, the rollback process is initiated to correct the status to ensure 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 approach 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 approach with that of a typical master server approach as a benchmark. The additional delay for data consistency of the proposed approach is reduced by 6.9-31.2$\%$ compared to the master server approach. In addition, the computation time was a few seconds, which is an acceptable time for network design before service launch. The proposed approach 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 approach is effective in guaranteeing data consistency with low delay.

Appendix A
Formulation of master server approach

In the master server method, one master server collates the status of all servers. The slave server sends the processing results to the master server. The master server is determined to minimize the delay with all servers. Figure 8(b) shows an example of the master server and slave servers when the server in Fig. 8(a) is selected. In the example in Fig. 8, 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
(Delay)
Link between master and slave servers
(Delay)
Link between server and server
Slave server
1
2
3
4
3
5
3
10
5
3
3
3
5

(a) Selected servers (b) Master and slave server pair

Fig. 8. Example of master and slave server pair for master server approach.

The parameters to be added and set in the Section III-B include $s_i, i \in V_{\mathrm{S}}$, is the server to be the master server if $s_i = 1$, and otherwise if $s_i = 0$. As variables, $m_i, i \in V_{\mathrm{S}}$, are binary variables, $m_i = 1$ if server i is elected as a master server, and $m_i = 0$ otherwise. An ILP problem as the server selection problem for the master server approach is given by:

$$\begin{aligned}
&\text{Objective} \quad \min \quad 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + 2L^{\max} && \text{(2a)}\\
&\text{s.t.} \quad m_i \leq y_i, \forall i \in V_{\mathrm{S}} && \text{(2b)}\\
&m_i + y_j - 1 \leq z_{ij}, \forall (i,j) \in E_{\mathrm{S}} && \text{(2c)}\\
&z_{ij} \leq m_i, \forall (i,j) \in E_{\mathrm{S}} && \text{(2d)}\\
&z_{ij} \leq y_j, \forall (i,j) \in E_{\mathrm{S}} && \text{(2e)}\\
&\sum_{i \in V_{\mathrm{S}}} s_i m_i = 1 && \text{(2f)}\\
&(1a) - (1l). && \text{(2g)}
\end{aligned}$$

Equation (2a) shows that the delay $2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} + 2L^{\max}$ is minimized as the objective function. The processing results are sent to the master server, and if the master server fails to collate the results, rollback instructions are sent to the master server to correct the processing results, so the collation delay is L_i round trip time, and $2L^{\max}$ is the maximum delay for data consistency. Equation (2b) indicates that the master server is elected from the servers to be selected. Equations (2c)-(2e) are linear expressions showing that $m_i y_j = z_{ij}$. Equation (2f) indicates that the selected server has one master server.

Appendix B
Formulation of server selection without data consistency

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

$$\begin{aligned}
&\text{Objective} \quad \min \quad 2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max} && \text{(3a)}\\
&(1a) - (1j). && \text{(3b)}
\end{aligned}$$

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 31 October 2022.

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

[3] 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,” *New York, John Wiley*, vol. 300, 2000.

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

[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.

[7] R. Baldoni and 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.

[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.

[9] “Japan Photonic Network Model,” https://www.ieice.org/cs/pn/eng/JPNM/TokyoMANModel.zip, online; accessed 31 October 2022.

[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] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, ”Local-lag and timewarp: providing consistency for replicated continuous applications,” *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 47-57, 2004.

[12] A. Kawabata, B. C. Chatterjee, and E. Oki, “An optimistic synchronization based optimal server selection scheme for delay sensitive communication services,” *IEICE Transactions on Communications*, vol. E104.B, no. 10, pp. 1277-1287, 2021.

[13] “IBM ILOG CPLEX,” https://www.ibm.com/analytics/cplex-optimizer, online; accessed 31 October 2022.