Optimistic Synchronization-Based Server Allocation with Preventive Start-Time Optimization for Delay-Sensitive Applications Under Server Failure

Masaki Oda†, Akio Kawabata‡, and Eiji Oki†

† Graduate School of Informatics, Kyoto University, Kyoto, Japan

‡ Toyohashi University of Technology, Aichi, Japan

IEEE RTC

07 – 08 October 2025 // Chicago, USA

Outline

- Background
- Related work and limitations
- Problem statement
- Proposed model
- NP-completeness
- Numerical Results
- Conclusions

Background

- Real-time applications (e.g., online gaming, ticket reservations) require low latency and strict event ordering.
- Traditional Centralized Processing (CP) ensures event order but causes significant delays, as it must wait for the slowest user.
- Distributed Server Processing (DSP) is effective for low latency.
- Synchronization algorithms are crucial for event ordering in DSP:
  - **Conservative Synchronization Algorithm (CSA)**
  - **Optimistic Synchronization Algorithm (OSA)**

Synchronization algorithms: CSA

- **Conservative Synchronization Algorithm (CSA)**
  - Enforces event order *before* processing.
  - Associates time information with each event.
- **DSP with CSA Model [Kawabata, IEEE, 2017]**
  - Step 1 (User → Server):
    $\leq D_{\mathrm{U}}^{\max}$
  - Step 2 (Server ↔ Server):
    $\leq D_{\mathrm{S}}^{\max}$ (Inter-server event sharing)
  - Step 3 (Server → User):
    $\leq D_{\mathrm{U}}^{\max}$
  - Total maximum delay:
    $2D_{\mathrm{U}}^{\max} + D_{\mathrm{S}}^{\max}$

[Kawabata, IEEE, 2017] 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, Dec. 2017.

Synchronization algorithms: OSA

- **Optimistic Synchronization Algorithm (OSA)**
  - Processes events *without* prior order enforcement.
  - If an out-of-order event is detected, the system performs a **rollback** to correct the state.
- **DSP with OSA Model [Kawabata, IEICE, 2021], [Kawabata, IEICE, 2020]**
  - Rollback requires servers to retain extensive historical state information.
  - This consumes more memory than CSA.
  - Total maximum delay:
    $2D_{\mathrm{U}}^{\max}$ (Round-trip delay between a user and their assigned server).
  - OSA can potentially improve delay performance compared to CSA.

DSP: distributed server processing CSA: conservative synchronization algorithm

[Kawabata, IEICE, 2021] 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, October 2021.
[Kawabata, IEICE, 2020] A. Kawabata, B. C. Chatterjee, and E. OKI, "Participating-domain segmentation based server selection scheme for real-time interactive communication," IEICE Trans. Commun., vol. E103.B, no. 7, pp. 736–747, 2020.

Server failures

- Servers can fail due to hardware malfunctions, software errors, or excessive load.
- Edge servers are generally more susceptible to failures than centralized cloud servers.
- When a server fails, users connected to it must be reassigned to alternative servers.
- Reallocation impacts the maximum total delay.
- This paper focuses on **single-server failures**, which are more prevalent than multiple failures.

Handling server failures: SO / RO

- **Start-time Optimization (SO)**
  - First, determines optimal assignment assuming no failures.
  - If a failure occurs, reallocates *only* users connected to the failed server.
  - *Pro:* Optimal when no failures occur.
  - *Con:* May not yield the lowest delay in failure scenarios.
- **Run-time Optimization (RO)**
  - Reassigns *all* users to available servers after a failure to minimize delay for that specific scenario.
  - *Pro:* Always achieves minimum delay for each failure scenario.
  - *Con:* May cause **instability** due to unnecessary reassignments for users whose servers are still operational.

Handling server failures: PSO

- **Preventive Start-time Optimization (PSO)**
  - Proactively assigns users to minimize the **largest total delay** across *all* possible failure scenarios.
  - To maintain stability such as SO: start-time optimization, PSO restricts the reassignment of users whose servers remain operational.
  - Effectively minimizes the worst-case delay while avoiding unnecessary service interruptions.

SO: start-time optimization

Related work (OSA [Kawabata, IEICE, 2021])

- The work in [Kawabata, IEICE, 2021] introduced an OSA-based server allocation model.

OSA: optimistic synchronization algorithm

- It considers the **Maximum Status Holding Time** constraint (H_{APL}, H_s).
- There is a trade-off between minimizing delay and satisfying the status holding time.
- **Example (Fig. 1):**
  - Allocation 1: Lower delay ($8 = 4 \times 2$), but higher status holding time ($14 = 4 + 10$).
  - Allocation 2: Higher delay ($14 = 7 \times 2$), but lower status holding time ($12 = 7 + 5$).
  - If $\min(H_{\mathrm{APL}} H_s) < 14$, Allocation 1 is infeasible.
- **Limitation:**
  This model does not consider server failures.

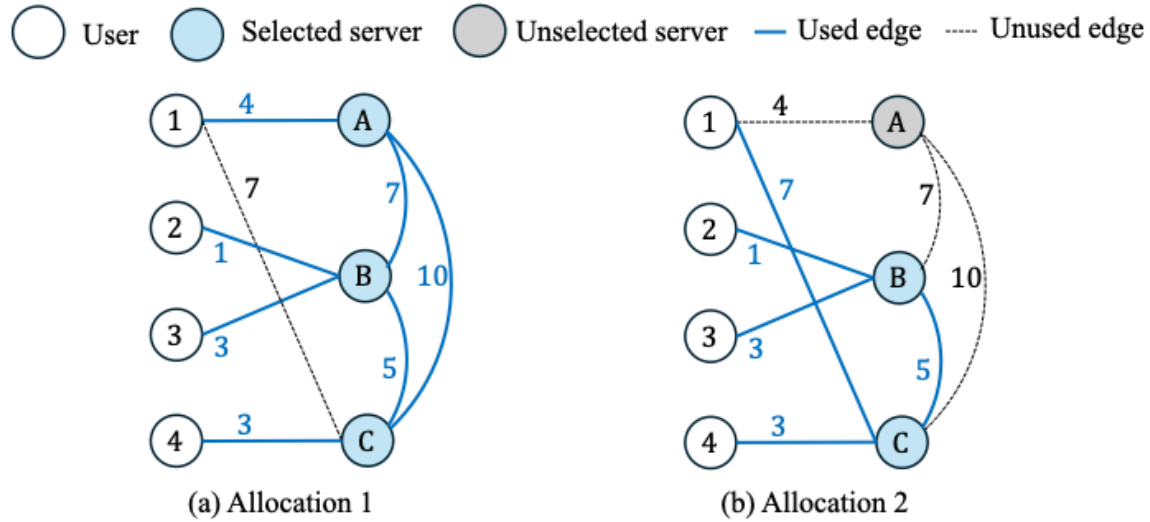


Fig. 1

[Kawabata, IEICE, 2021] 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, October 2021.

Related work (CSA+PSO [Masuda, HPSR, 2020])

- The work in [Masuda, HPSR, 2020] addressed a CSA-based DSP model with PSO under single-server failures.
- **Example (Fig. 2):**
  - **SO:** Optimal at $f = 0$ ($D^0_{\mathrm{SO}} = 15$), but suboptimal in failure scenarios ($D_{\mathrm{SO}} = 18$).
  - **RO:** Optimal in every scenario ($D_{\mathrm{RO}} = 16$), but causes instability (e.g., user 3 reassigned at $f = 1$ even though server 2 is non-failed).
  - **PSO:** Balances stability and worst-case delay ($D_{\mathrm{PSO}} = 16$).

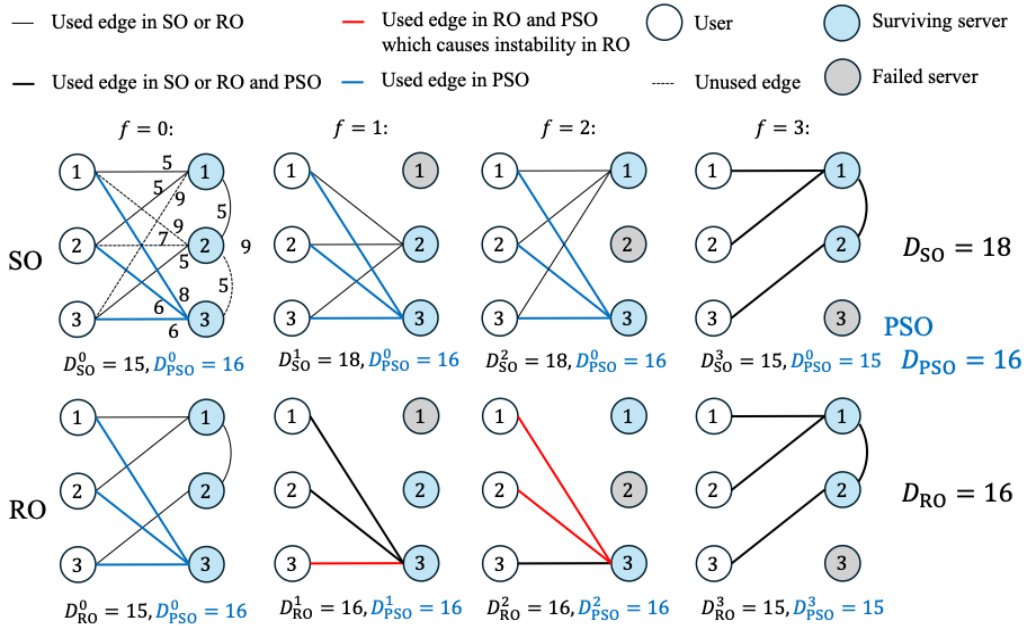


Fig. 2

- **Limitation:** This model does not incorporate OSA's constraints (e.g., maximum status holding time).

PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization
DSP: distributed server processing
CSA: conservative synchronization algorithm
OSA: optimistic synchronization algorithm

[Masuda, HPSR, 2020] S. Masuda, F. He, A. Kawabata, and E. Oki, "Distributed server allocation model with preventive start-time optimization against single failure," in IEEE 21st Int. Conf. High Performance Switching and Routing (HPSR), 2020, pp. 1–6.

Problem statement

- OSA-based server allocation in the presence of server failures remains insufficiently investigated.

- A question arises:
  - *Can we develop a model for OSA-based server allocation with PSO against single-server failures?*

PSO: preventive start-time optimization　OSA: optimistic synchronization algorithm

Proposed model: OSA-based server allocation with PSO

- This paper proposes an **OSA-based server allocation model with PSO** against single-server failures **(OSA-PSO)**.
  - The model minimizes the largest total delay across all failure scenarios while satisfying **maximum status holding time constraints**.
  - The model ensures stability by preventing unnecessary user reassignments.

- We formulate the model as an **integer linear programming (ILP)** problem.
- We prove that the decision version of the problem is **NP-complete**.

PSO: preventive start-time optimization
OSA: optimistic synchronization algorithm
ILP: integer linear programming

Proposed model: OSA-PSO (definitions)

- **Given parameters:**
  - V_{U} : Set of users
  - V_{S} : Set of potential server locations
  - M_s : Server capacity
  - F: Set of failure scenarios ($f = 0$ for no failure, $f \in V_{\mathrm{S}}$ for server f failure)
  - $d(u, s), d(s, t)$: Delays
  - H_{APL}, H_s : Maximum status holding times
- **Decision parameters:**
  - x_{ijf}: Binary. 1 if edge (i, j) is used in scenario f.
  - y_{sf} : Binary. 1 if server s is selected in scenario f.
  - $D_{\mathrm{U}f}$: Largest $d(u, s)$ used in scenario f.
- **Objective:**
  - Minimize D, the largest total delay $2D_{\mathrm{U}f}$ over all failure scenarios $f \in F$.

PSO: preventive start-time optimization OSA: optimistic synchronization algorithm

Proposed model: OSA-PSO (ILP formulation)

- **Objective:** min D
- **Subject to (Major Constraints):**
  - $\sum_{s \in V_S \setminus \{f\}} x_{usf} = 1, \forall u, f$
    - Each user connects to exactly one non-failed server.
  - $\sum_{u \in V_U} x_{usf} \leq M_s, \forall s, f$
    - Server capacity constraint.
  - $x_{usf} d(u,s) \leq D_{Uf}, \forall (u,s), f$
    - Calculate max user-server delay D_{Uf}.
  - $2D_{Uf} \leq D, \forall f$
    - Calculate largest total delay D.
  - $x_{us0} \leq x_{usf}, \forall f \neq 0, u, s \neq f$
    - **PSO stability constraint**: Users connected to non-failed servers are not reassigned.
  - $x_{usf} d(u,s) + x_{stf} d(s,t) \leq \min(H_{APL}, H_s), \forall s, (u,s), (s,t), f$
    - **Maximum status holding time constraint**.

PSO: preventive start-time optimization
OSA: optimistic synchronization algorithm
ILP: integer linear programming

NP-completeness

- **Definition 1 (D-OSA-PSO):**
  - The decision version of the problem: Is there a feasible allocation such that the largest total delay $D \leq h$, while satisfying all constraints (including H_{APL}, H_s)?
- **Theorem 1:** D-OSA-PSO is NP-complete.
- **Proof outline:**
  - **D-OSA-PSO is in NP:** A given solution (allocation) can be verified in polynomial time $O(|V_{\mathrm{U}}||V_{\mathrm{S}}|^2)$.
  - **NP-hard:** We prove this by polynomial-time reduction from **3-SAT** (a known NP-complete problem).
  - We construct an instance J of D-OSA-PSO from any instance I of 3-SAT (Fig. 3).
  - The 3-SAT instance I is satisfiable **if and only if** the D-OSA-PSO instance J is "yes" (i.e., $D \leq h$).

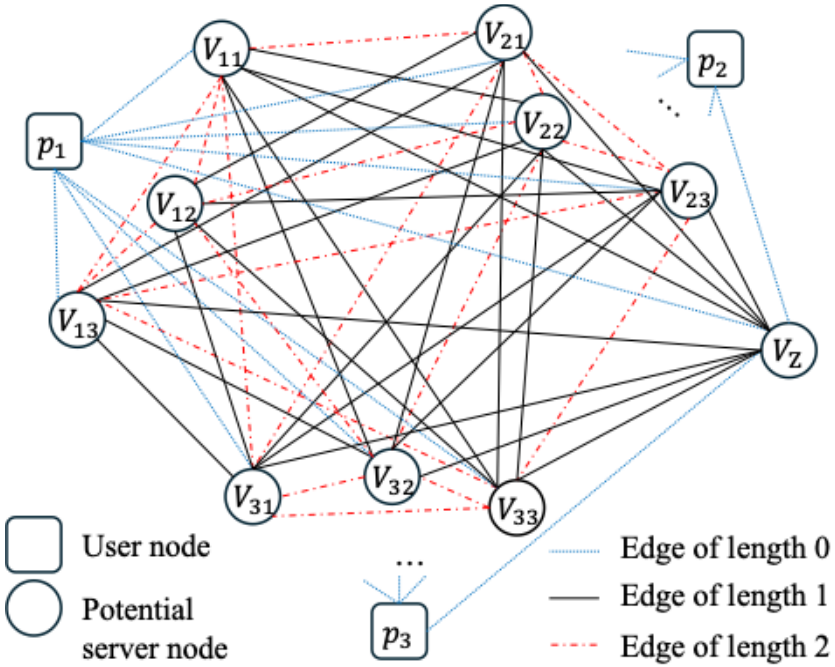
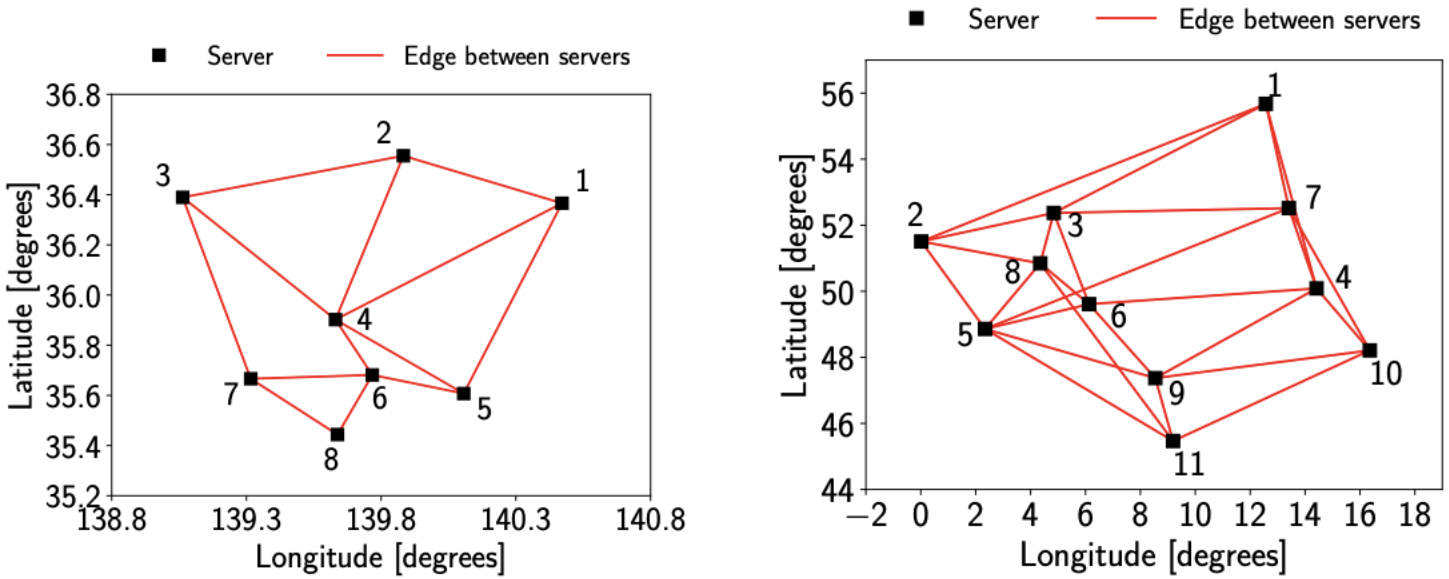


Fig. 3

PSO: preventive start-time optimization OSA: optimistic synchronization algorithm

Numerical results (setup)

- **Environment:**
  - PC: Intel Core i7-7700 3.60 GHz, 32 GB RAM
  - ILP solver: CPLEX 12.10.0.0
- **Network topologies (Fig. 4):**
  - JPN Kanto (8 servers)
  - COST 239 (11 servers)
- **Experiment settings:**
  - Users ($|V_U|$): 30 to 70
  - Server Capacity (M_s): Set to $|V_U|$ (i.e., no capacity limit) to observe basic characteristics.
  - Trials: 20 independent trials for each setting.
  - Delay Calculation: 0.5 ms per 100 km.



(a) JPN Kanto (b) COST 239

Fig. 4

ILP: integer linear programming

Impact of H_{APL} (vs CSA)

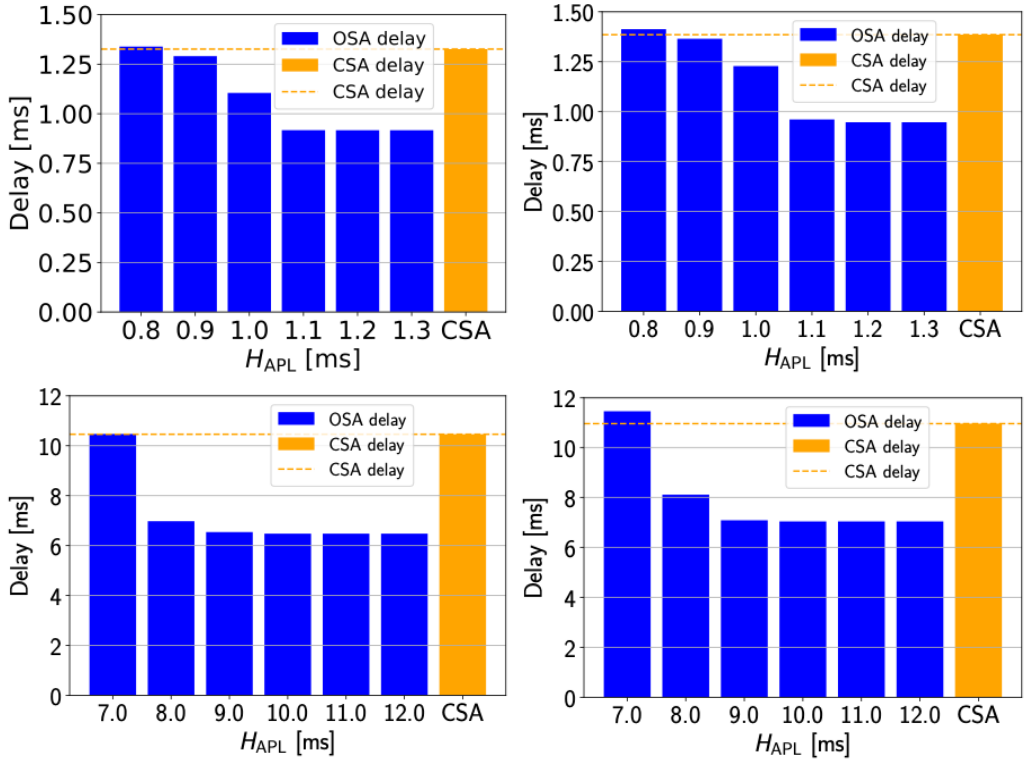
PSO: preventive start-time optimization
CSA: conservative synchronization algorithm
OSA: optimistic synchronization algorithm

OKILAB
Intelligent Communication Networks Lab

- **Comparison:** Proposed OSA-PSO vs. conventional CSA-PSO.
- **Setting:** $H_s = H_{\mathrm{APL}}$.
- **Results (JPN Kanto):**
  - OSA delay decreases as H_{APL} increases.
  - When $H_{\mathrm{APL}} \geq 0.9$ ms,
    OSA achieves lower delay than CSA.
- **Results (COST 239):**
  - When $H_{\mathrm{APL}} \geq 8.0$ ms,
    OSA achieves lower delay than CSA.
- **Observation:**
  - When H_{APL} is small, the status holding constraint forces users to select more distant servers, increasing OSA's delay.
  - When H_{APL} is large, OSA-PSO is effective in reducing delay.

JPN

COST



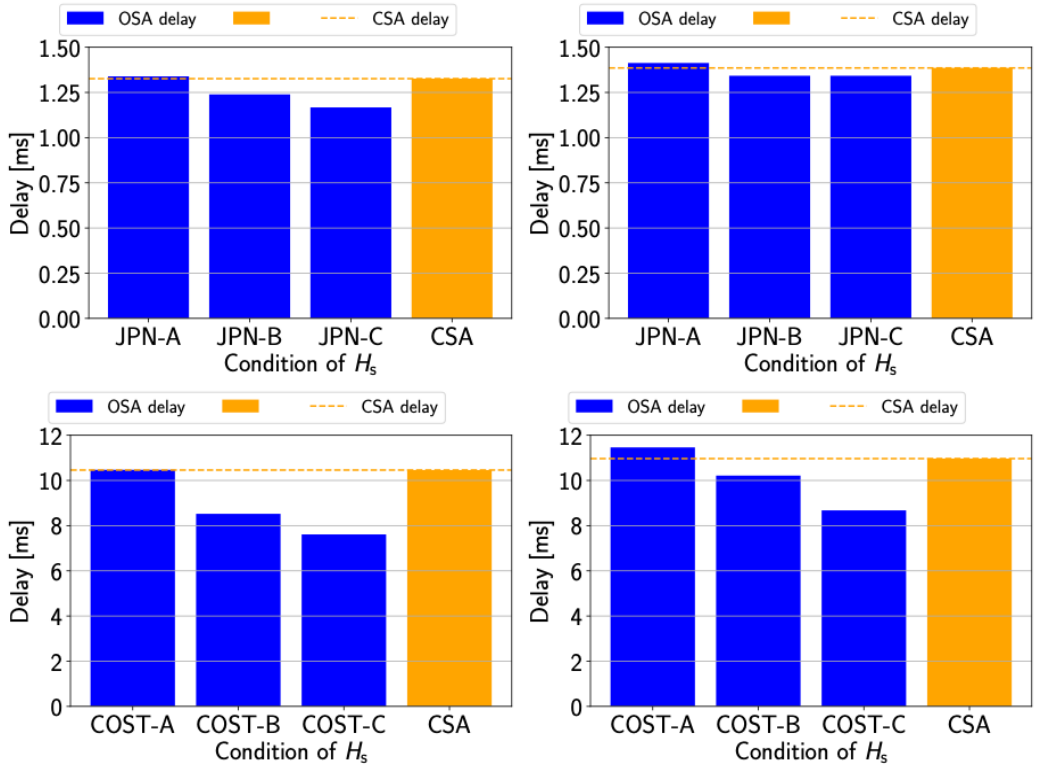
30 users 70 users

Impact of H_s (vs CSA)

PSO: preventive start-time optimization
CSA: conservative synchronization algorithm
OSA: optimistic synchronization algorithm

- **Objective:** Investigate if increasing servers with larger memory (higher H_s) reduces delay.
- **Setting:** $H_{\mathrm{APL}} = \max(H_s)$. H_s values are varied.
  - e.g., JPN-A (all low H_s) → JPN-C (more high H_s)
- **Results (JPN Kanto & COST 239):**
  - Delay decreases as the number of servers with greater memory resources (H_s) increases.
  - (e.g., Delay of JPN-A > JPN-B > JPN-C)
- **Observation:**
  - The proposed OSA-PSO model effectively utilizes server memory resources to minimize delay.

JPN-A	$0.8,\ \forall s \in V_S$	[ms]
JPN-B	$0.8\,(s = 1,3,4,5,7,8),\ 1.0\,(s = 2,6)$	
JPN-C	$0.8\,(s = 1,3,5,7),\ 1.0\,(s = 2,4,6,8)$	
COST-A	$7.0,\ \forall s \in V_S$	[ms]
COST-B	$7.0\,(s = 1,2,3,4,5,7,8,9,10,11),\ 9.0\,(s = 6,8)$	
COST-C	$7.0\,(s = 1,3,5,7,8,9,10,11),\ 9.0\,(s = 2,4,6,8)$	



30 users 70 users

PSO vs SO/RO (metrics)

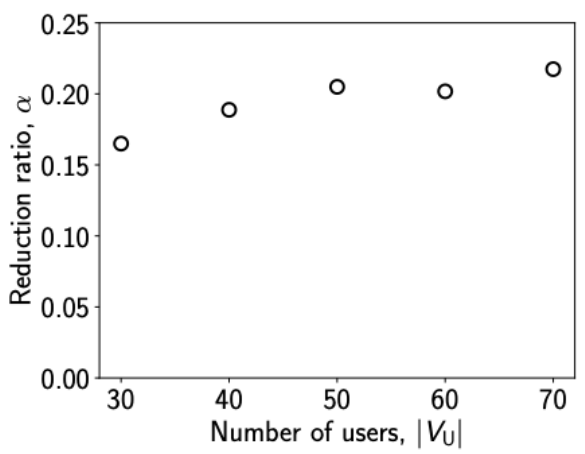
- **Objective:** Compare proposed OSA-PSO with baselines SO and RO.
- **Metrics [Masuda, HPSR, 2020], [Kamrul, IEEE, 2010]:**
  - α: Reduction ratio of PSO vs. SO (Advantage of PSO).
    - $\alpha = (\max D^f_{\mathrm{SO}} - D_{\mathrm{PSO}})/\max D^f_{\mathrm{SO}}$
  - β: Penalty of PSO vs. SO at $f = 0$ (Cost of PSO).
    - $\beta = (D^0_{\mathrm{PSO}} - D^0_{\mathrm{SO}})/D^0_{\mathrm{SO}}$
  - γ: Increase ratio of PSO vs. RO (Delay penalty of PSO).
    - $\gamma = (D_{\mathrm{PSO}} - \max D^f_{\mathrm{RO}})/\max D^f_{\mathrm{RO}}$
  - ν: Average ratio of unnecessary disconnections in RO (Instability of RO).

PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization
OSA: optimistic synchronization algorithm

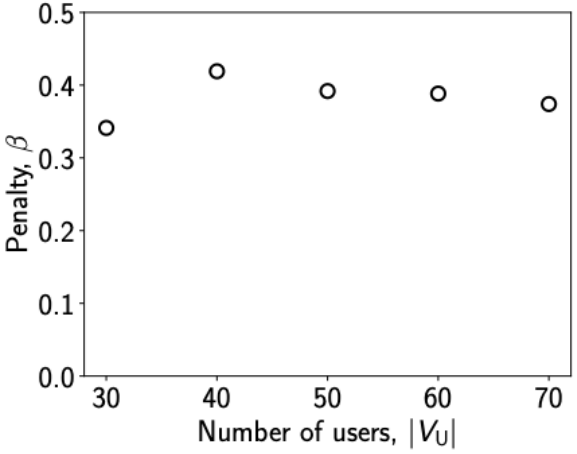
[Masuda, HPSR, 2020] S. Masuda, F. He, A. Kawabata, and E. Oki, "Distributed server allocation model with preventive start-time optimization against single failure," in IEEE 21st Int. Conf. High Performance Switching and Routing (HPSR), 2020, pp. 1–6.
[Kamrul, IEEE, 2010] I. M. Kamrul and E. Oki, "PSO: preventive start-time optimization of OSPF link weights to counter network failure," IEEE Commun. Lett., vol. 14, no. 6, pp. 581–583, 2010.

PSO vs SO/RO (JPN Kanto)

- **Setting:** $H_{\mathrm{APL}} = H_s = 1.1$ ms.
- **PSO vs SO:**
  - α (Advantage):
    PSO achieves significant delay reduction.
    Average α is 0.20 (20%).
    Max reduction in one trial was 0.41 (41%)
  - β (Penalty):
    PSO has a penalty at $f = 0$. Average β is 0.38 (38%).
- **PSO vs RO:**
  - γ (Delay penalty): PSO's delay is nearly identical to RO.
    γ is very small (avg. 4.3×10^{-4}).
  - ν (Instability): RO causes $42 - 48\%$ of users to be unnecessarily disconnected per failure. PSO has $\nu = 0$.



(a) α



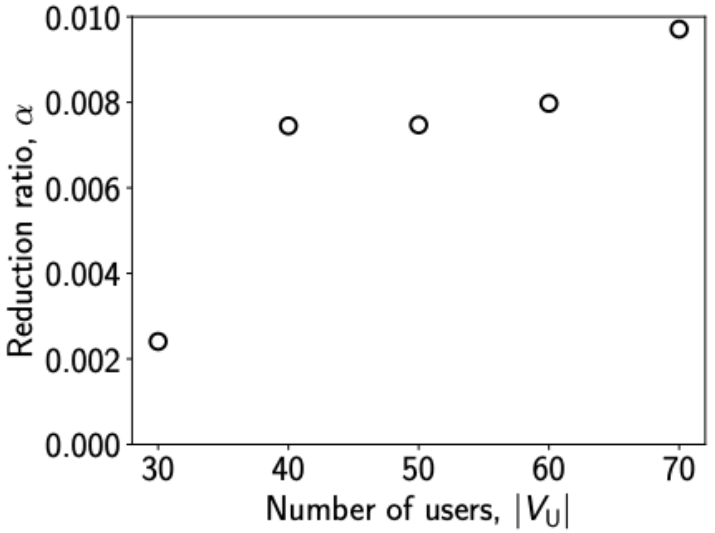
(b) β

$\lvert V_{\mathrm{U}} \rvert$	JPN Kanto	
	γ	ν
30	3.8×10^{-4}	0.42
40	4.9×10^{-4}	0.43
50	4.9×10^{-4}	0.44
60	3.8×10^{-4}	0.47
70	3.8×10^{-4}	0.48

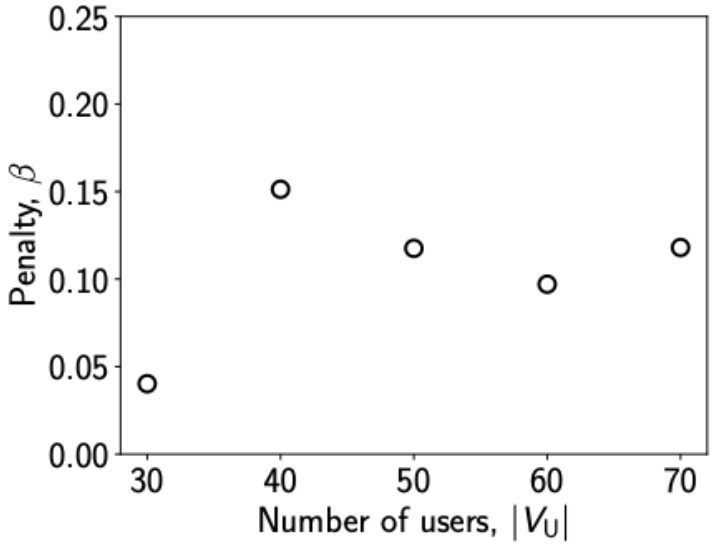
PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization

PSO vs SO/RO (COST 239)

- **Setting:** $H_{\mathrm{APL}} = H_s = 10$ ms.
- **PSO vs SO:**
  - α (Advantage):
    PSO shows delay reduction,
    especially as users increase.
  - β (Penalty):
    PSO has a penalty at $f = 0$.
- **PSO vs RO:**
  - γ (Delay penalty):
    PSO's delay is identical to RO ($\gamma = 0.0$).
  - ν (Instability):
    RO causes $34 - 41\%$ of users
    to be unnecessarily disconnected.
    PSO has $\nu = 0$.



(a) α



(b) β

	COST 239	
$\lvert V_{\mathrm{U}}\rvert$	γ	ν
30	0.0	0.34
40	0.0	0.37
50	0.0	0.40
60	0.0	0.37
70	0.0	0.41

PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization

Discussion of PSO vs SO/RO

- **PSO vs. SO:**
  - PSO achieves a significantly lower *largest total delay* (worst-case) than SO.
  - Average reduction (α) is 10%, with a maximum of 41%.
  - This comes at the cost of a slightly higher delay in the no-failure scenario (β).
- **PSO vs. RO:**
  - PSO achieves a largest total delay that is **nearly equivalent** to RO (the optimal for each scenario).
  - PSO **completely eliminates instability** ($\nu = 0$).
  - RO causes significant instability, unnecessarily reassigning 41% of users on average per failure.

PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization

Conclusions

- We proposed an **optimistic synchronization-based server allocation model with PSO (OSA-PSO)** against single-server failures.
  - The model minimizes the largest total delay across all failure scenarios while satisfying maximum status holding time constraints.
- We formulated the model as an ILP problem.
- We proved its decision version is **NP-complete**.
- **Numerical results show:**
  - OSA-PSO achieves **lower delay than CSA-PSO** when memory (status holding time) is sufficient.
  - OSA-PSO **outperforms SO** in worst-case delay (10% avg. reduction, 41% max).
  - OSA-PSO **matches RO's delay performance** while **eliminating RO's instability** (41% avg. unnecessary disconnections).

PSO: preventive start-time optimization
SO: start-time optimization
RO: run-time optimization
CSA: conservative synchronization algorithm
OSA: optimistic synchronization algorithm
ILP: integer linear programming