

Protocol-Aware Dynamic Synchronization for Multi-Rate Maritime Co-Simulation

Woo-Sung Jung, Jeongsik Kim, Dae Seung Yoo

Ulsan Intelligent Convergence Research Section

Electronics and Telecommunications Research Institute

Republic of Korea

woosung@etri.re.kr, j.s.kim@etri.re.kr, ooseyds@etri.re.kr

Abstract—Maritime cyber-physical systems increasingly integrate navigation, control, sensing, and heterogeneous communication technologies, requiring simulation frameworks that accurately capture interactions across widely different time scales. However, conventional co-simulation approaches with a uniform communication step fail to represent the millisecond-level timing of transport-protocol events while simultaneously accommodating the second-level evolution of vessel motion and control processes. This paper proposes a protocol-aware dynamic synchronization method for multi-rate maritime co-simulation. The method leverages internal protocol-state information to switch between fast and slow execution modes, selectively activates only the FMUs relevant to each mode, and emulates event-driven actuator activation without requiring advanced clock semantics. Numerical experiments demonstrate that the proposed method restores realistic TCP handshake latency under varying network delay and packet-loss conditions while significantly reducing the computational load associated with slow-scale physical models, achieving the fidelity of fine-step simulations at a fraction of the cost.

Index Terms—Co-simulation, Dynamic Synchronization, FMI

I. INTRODUCTION

The Functional Mock-up Interface (FMI) has become a widely adopted standard for the co-simulation of heterogeneous cyber-physical systems (CPS) [1], [2]. With the increasing digitalization and autonomy of maritime operations—exemplified by the ongoing discussion on Maritime Autonomous Surface Ships (MASS) in the International Maritime Organization (IMO) [3]—there is a growing interest in integrating ship dynamics, communication networks, remote control algorithms, and cloud-based decision modules into a unified simulation environment. FMI enables modular integration across such domains while preserving the encapsulation of domain-specific solvers.

A central challenge in maritime co-simulation is that the constituent subsystems operate on fundamentally different temporal scales. Communication modules that model TCP-like handshakes, transport-layer retransmission mechanisms, or gateway behavior towards satellite links may require millisecond or even sub-millisecond resolution, because the exact timing of protocol events directly influences message delays and control-loop responsiveness. In contrast, ship dynamics evolve at time scales of seconds due to large mass and inertia, and actuator modules may require event-triggered execution only when command packets arrive.

At the same time, not all maritime communication services are designed for low-latency interaction. Narrowband systems such as the VHF Data Exchange System (VDES) [4] and satellite-based maritime broadband links often operate with relatively low data rates and substantial propagation delay; geostationary satellite paths, for example, naturally exhibit round-trip times on the order of several hundred milliseconds [5]. For slow telemetry, monitoring, or periodic reporting, a coarse effective sampling period is acceptable and there is little benefit in driving the communication FMU at a sub-millisecond step size. However, the *onset* of critical events—such as the transfer of control to a remote operator, the initiation of a collision-avoidance maneuver, or the delivery of fail-safe commands—still requires fast detection and reaction at the scale of the transport protocol. Thus, a realistic maritime co-simulation must support short communication steps during such bursts of activity, even if the average traffic pattern is sparse or high-latency.

Traditional FMI-based co-simulation typically relies on a uniform global communication step. However, this assumption becomes untenable for multi-rate maritime systems. If the global step is large (e.g., 1 s), packet exchanges are artificially delayed and TCP handshake latency becomes unrealistically inflated, masking the true impact of RTT and packet loss on control performance. If the step is small (e.g., 1 ms), slow FMUs such as ship-dynamics must be called at an excessively high frequency, resulting in substantial computational overhead and limiting the scalability to multi-ship scenarios.

Therefore, a new synchronization strategy is required—one that can dynamically adapt to communication patterns, activate only the relevant FMUs at the required time scale, and avoid wasteful updates of slow FMUs. This paper addresses this need by proposing a communication-aware dynamic synchronization framework that enables realistic co-simulation of ship-shore TCP communication and slow-scale ship-motion FMUs within existing FMI 2.0 infrastructures, without requiring full migration to FMI 3.0 or specialized discrete-event extensions.

II. RELATED WORK

A. FMI Co-Simulation and Its Limitations

FMI 2.0 Co-Simulation is widely used to integrate heterogeneous simulators, where each unit is advanced by master-issued `doStep` calls [1]. However, its two structural con-

straints pose challenges for systems that mix fast communication processes with much slower physical dynamics. First, FMI 2.0 enforces a single global communication step, forcing all components to update on the same discrete time grid regardless of their natural time scales. Second, the standard lacks explicit clock or event semantics, so event-driven behavior must be emulated through frequent polling, which becomes inefficient when only a subset of components requires high-frequency updates.

These limitations become critical in maritime scenarios where protocol events (e.g., TCP transitions, retransmissions, or gateway behavior toward VDES or satellite links) occur at millisecond scale, while physical dynamics evolve slowly and communication traffic may be sparse or highly delayed. A realistic simulation must support short steps during bursty communication activity and long steps during idle periods—capabilities that the FMI 2.0 execution model does not provide directly.

Despite these drawbacks, FMI 2.0 remains the dominant interface in maritime digitalization efforts. The Open Simulation Platform (OSP), widely adopted for ship and system-of-systems verification, relies on FMI 2.0 and hosts a large ecosystem of existing models and tools [6]. Migrating these assets to FMI 3.0 is costly and non-trivial, due to differences in clock semantics and hybrid interfaces.

Thus, practical maritime tool chains continue to require synchronization strategies that operate within FMI 2.0 while supporting realistic multi-rate communication–dynamics coupling. The method proposed in this paper addresses this need by enabling protocol-aware, dynamically adaptive execution without modifying existing FMI 2.0/OSP models.

B. Multi-rate Co-simulation

Co-simulation surveys such as Gomes *et al.* [2] and Hafner *et al.* [7] classify master algorithms, interpolation schemes, and numerical issues in multi-domain simulation. These works highlight multi-rate scheduling, where fast and slow components use different step sizes, often coordinated through extrapolation or interpolation. Palensky *et al.* [8] emphasize the importance of multi-rate frameworks for cyber–physical energy systems, in which ICT events occur faster than electrical or mechanical states.

However, these approaches generally rely on *numerical criteria* (local error, stability bounds) or *static configuration*. They do not consider *protocol-level* semantics, such as TCP state transitions, retransmissions, or handshake phases, to drive dynamic step-size selection. Thus, they cannot detect when fine-grained communication timing is required, nor do they reduce computation when communication is idle.

C. Event-based and Hybrid Synchronization

FMI 3.0 introduces explicit clocks and event-triggered activation, enabling masters to advance only the FMUs whose clocks tick [9]. This greatly improves synchronization for hybrid systems that mix continuous dynamics and discrete events.

However, most existing maritime tool chains remain tied to FMI 2.0, including commercial naval simulators, communication-system FMUs, and ship-control testing platforms. Under FMI 2.0, there is no native support for event-triggered communication; the master must repeatedly call `doStep` at a short step size to “discover” communication events. This produces unnecessary updates to slow FMUs when the communication FMU is the only component requiring high-frequency attention.

D. Co-simulation of Communication and Dynamics

Communication-aware co-simulation has been studied mainly in smart grids and automated vehicles. Palensky *et al.* [8] show that coupling power dynamics with ICT simulators is essential to evaluate latency and loss effects. Rehtanz *et al.* [10] demonstrate that realistic ICT delays significantly affect wide-area protection systems. Kim *et al.* [11] show that co-operation approach with heterogeneous network simulator by employing a standardized interface and open-source tools.

These studies confirm that combining fast communication and slow physical dynamics is necessary, but they do not address:

- maritime-specific communication characteristics (e.g., VDES low bitrate, satellite RTT 300–800 ms, bursty control events),
- FMI-specific constraints such as lack of event clocks in FMI 2.0,
- protocol-phase-dependent execution, where only the communication FMU should run at high rate.

Thus existing approaches cannot provide a mechanism for communication FMUs to trigger dynamic fast-mode execution while allowing ship-dynamics FMUs to remain in slow mode.

E. Gap Addressed in This Work

Across the literature, there is a clear recognition that heterogeneous time scales must be supported, but there is no FMI 2.0-compatible method that:

- leverages the *internal protocol state* of a communication FMU (e.g., SYN, SYN/ACK, ACK, timeout, retransmission) to drive synchronization,
- dynamically reconfigures the active FMU set (fast vs. slow clusters) during simulation,
- preserves millisecond-scale timing accuracy for communication **without** forcing ship-dynamics FMUs into millisecond-step evaluations,
- explicitly targets maritime communication environments, where *bursty short events occur on top of high-latency, low-rate channels*.

The present work fills this gap by proposing a communication-aware dynamic synchronization algorithm that: (1) detects communication activity via protocol-state variables, (2) switches execution to fast-mode only during communication-critical intervals, and (3) holds slow FMUs using zero-order hold to avoid unnecessary computation.

III. PROPOSED METHOD

This section presents the proposed communication-aware dynamic synchronization method for FMI-based co-simulation of maritime systems. The key idea is to exploit protocol-level information from the communication FMU to (i) dynamically switch between fast and slow execution modes and (ii) selectively activate only the FMUs that are relevant at each mode, while holding the state of slow FMUs via zero-order hold.

A. Cluster-Based Partitioning of FMUs

We explicitly partition the FMUs into two clusters as shown in Fig. 1:

$$\mathcal{C}_{\text{fast}} = \{A, B, C, (E \text{ when commanded})\}, \quad (1)$$

$$\mathcal{C}_{\text{slow}} = \{D\}. \quad (2)$$

We denote the fast and slow communication step sizes by H_{fast} and H_{slow} , respectively, with $H_{\text{fast}} \ll H_{\text{slow}}$ (e.g., $H_{\text{fast}} = 1$ ms, $H_{\text{slow}} = 1$ s). The master algorithm will selectively advance either $\mathcal{C}_{\text{fast}}$ or $\mathcal{C}_{\text{slow}}$ depending on the current communication activity.

B. Protocol-Aware Mode Switching

The Communication FMU B exports a Boolean signal

$$\text{active} \in \{\text{true}, \text{false}\}, \quad (3)$$

which indicates whether TCP handshake or packet exchange is currently ongoing. This signal is derived from the internal protocol state machine of the Communication FMU (e.g., SYN-SYN/ACK-ACK, retransmissions, or active data transfer).

The master uses `active` to select the execution mode:

$$(H, \mathcal{C}_{\text{act}}) = \begin{cases} (H_{\text{fast}}, \mathcal{C}_{\text{fast}}), & \text{if } \text{active} = \text{true}, \\ (H_{\text{slow}}, \mathcal{C}_{\text{fast}} \cup \mathcal{C}_{\text{slow}}), & \text{otherwise,} \end{cases} \quad (4)$$

where H denotes the step size for the current macro-step and $\mathcal{C}_{\text{act}}$ the set of FMUs to be advanced in that step as shown in Fig. 2.

C. Zero-Order Hold for Slow Dynamics

When the system is in fast mode, the ship-dynamics FMU D is not advanced in every fast step. Instead, its latest state

$$\mathbf{x}_D(t_k) \quad (5)$$

computed at the last slow step t_k is held constant for $t \in [t_k, t_{k+1})$ and provided to FMUs A and E as a zero-order hold (ZOH) approximation:

$$\mathbf{x}_D(t) \approx \mathbf{x}_D(t_k), \quad t_k \leq t < t_{k+1} = t_k + H_{\text{slow}}. \quad (6)$$

This approximation is justified because ship motion changes negligibly over millisecond intervals compared to the second-level dynamics.

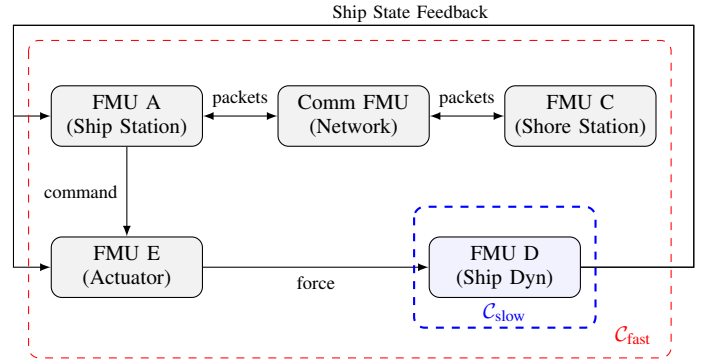


Fig. 1. FMU partitioning into fast communication and slow dynamics clusters.

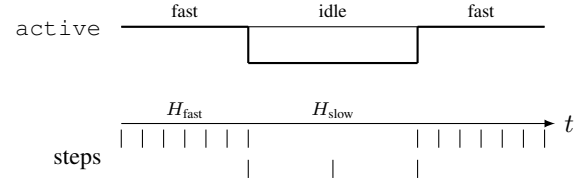


Fig. 2. Fast/slow mode switching controlled by `comm_active`.

D. Event-Driven Activation of the Actuator FMU

The actuator FMU E is activated only when the ship-control FMU A issues a new command. FMU A outputs a Boolean flag `command_valid` along with the command payload. The master interprets this signal as a pseudo-clock in an FMI 2.0 environment and only calls FMU E when

$$\text{command_valid} = \text{true}. \quad (7)$$

This emulates the event-triggered behavior of FMI 3.0 clocks without requiring FMI 3.0 support.

E. Algorithm Description

Algorithm 1 summarizes the proposed communication-aware synchronization strategy. The master operates by selecting, at every macro-step, the appropriate time step size and the subset of FMUs that must be advanced.

At each iteration, the master queries the Communication FMU B to obtain the Boolean flag `active`, which reflects whether the underlying transport protocol is currently engaged in handshake, retransmission, or data exchange. This signal drives the choice between fast and slow execution modes.

When `active` is true, the master enters fast mode: it uses the fine step size H_{fast} and advances only the fast cluster $\mathcal{C}_{\text{fast}}$. The slow ship-dynamics FMU D is not evaluated in this phase; its most recent state is supplied to the fast FMUs using a zero-order hold, enabling millisecond-level communication fidelity without the computational cost of updating slow dynamics.

When `active` becomes false, the master switches to slow mode. The step size increases to H_{slow} , and both the fast and slow clusters are advanced. FMU D is evaluated only at these slow-mode boundaries, and its updated state becomes the new ZOH value for subsequent fast-mode intervals.

Algorithm 1 Communication-Aware Dynamic Synchronization Master

```

1: Initialize all FMUs and set  $t \leftarrow 0$ .
2: Obtain initial ship state  $\mathbf{x}_D$  from FMU D.
3: while  $t < T_{\text{end}}$  do
4:   Read active from FMU B.
5:   if active = true then ▷ Fast mode
6:      $H \leftarrow H_{\text{fast}}; \mathcal{C}_{\text{act}} \leftarrow \mathcal{C}_{\text{fast}}$ .
7:     Provide held  $\mathbf{x}_D$  to FMUs A and E.
8:   else ▷ Slow mode
9:      $H \leftarrow H_{\text{slow}}; \mathcal{C}_{\text{act}} \leftarrow \mathcal{C}_{\text{fast}} \cup \mathcal{C}_{\text{slow}}$ .
10:  end if
11:  for all FMUs  $F \in \mathcal{C}_{\text{act}}$  do
12:    Call doStep( $F, t, H$ ).
13:  end for
14:  Read command_valid and command from FMU A.
15:  if command_valid = true then
16:    Advance FMU E with the new command.
17:    Update the force/moment input to FMU D.
18:  end if
19:  if comm_active = false then
20:    Update  $\mathbf{x}_D$  from FMU D at the slow step.
21:  end if
22:   $t \leftarrow t + H$ .
23: end while

```

In addition to mode switching, the master supports event-driven activation of the actuator FMU E. Whenever the ship-control FMU A asserts the `command_valid` flag, FMU E is executed immediately, regardless of the current mode. This behavior effectively emulates clocked execution from FMI 3.0 while remaining compatible with FMI 2.0.

In summary, the algorithm performs high-frequency co-simulation only when communication activity demands it, while minimizing unnecessary evaluations of slow FMUs. This protocol-aware scheduling differs fundamentally from uniform-step or purely numerically driven multi-rate approaches by exploiting explicit protocol state to dynamically reconfigure the active FMU set.

IV. PERFORMANCE EVALUATION

This section evaluates the proposed communication-aware synchronization method using five groups of experiments: (i) baseline comparison of handshake latency, (ii) FMU D invocation count, (iii) sensitivity to network round-trip time (RTT), (iv) impact of packet loss under low RTT (10 ms), and (v) impact of packet loss under high RTT (100 ms).

All experiments simulate a TCP-like three-way handshake followed by a command transmission from the shore station controller (FMU C) to the ship station controller (FMU A), which then triggers the actuator FMU E and affects the slow ship-dynamics FMU D. Three synchronization strategies are compared:

- **Global-1s:** a uniform global macro-step $H = 1$ s.
- **Global-1ms:** a uniform step $H = 1$ ms.

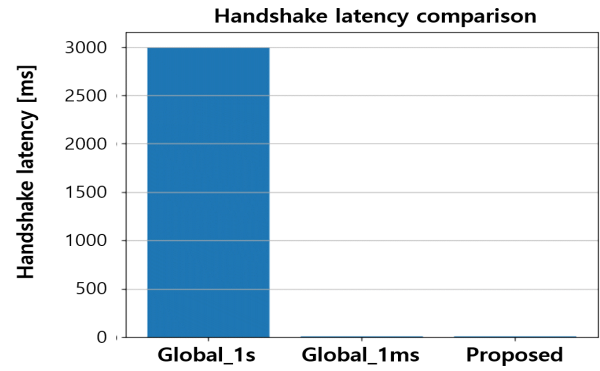


Fig. 3. Handshake Latency Comparison

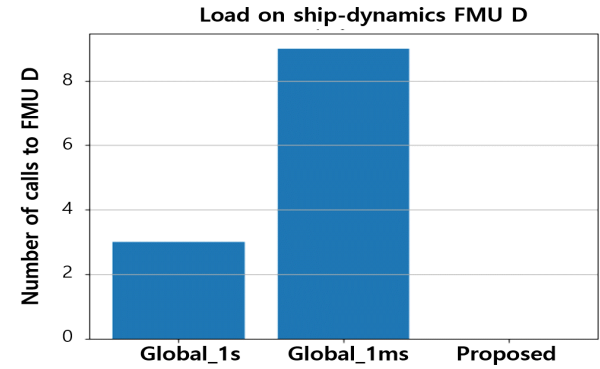


Fig. 4. Loads on Slow FMU

- **Proposed:** dynamic clustering with $H_{\text{fast}} = 1$ ms and $H_{\text{slow}} = 1$ s.

A. Baseline Comparison: Handshake Latency

Fig.3 compares the completion time of the SYN-SYN/ACK-ACK handshake. As expected, Global-1s produces an inflated latency of approximately 3 s because each packet event is quantized to the 1-second step boundary. Global-1ms eliminates quantization error and achieves a realistic millisecond-scale handshake. The proposed method achieves nearly identical latency to Global-1ms because it enters fast mode only during active packet exchange. This confirms that protocol timing fidelity can be preserved without globally applying a fine step size.

B. Invocation Count of FMU D

Fig.4 reports the number of calls to the slow ship-dynamics FMU D over a 10-second simulation. Global-1ms calls FMU D at every millisecond, producing more than 9,000 invocations and rendering it computationally impractical for high-fidelity vessel models. In contrast, Global-1s and the proposed method invoke FMU D only during slow-mode intervals. The proposed method thus achieves the same realism in communication timing as Global-1ms while maintaining the computational load close to Global-1s—demonstrating the key benefit of communication-driven clustering.

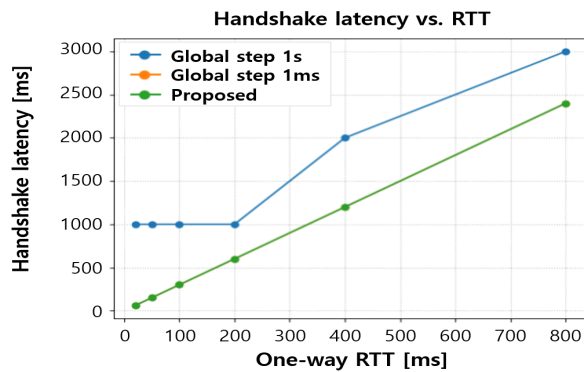


Fig. 5. Handshake Latency

C. Effect of Increased RTT

To evaluate behavior under maritime network conditions, Fig.5 illustrates handshake latency as a function of one-way RTT from 50 to 800 ms.

- For low RTT (50–200 ms), Global-1s remains bounded at ≈ 1 s, dominated by macro-step quantization rather than physical delay.
- For $RTT \geq 400$ ms, Global-1s begins reflecting the actual network delay, but still with large step-level distortions.
- Global-1ms accurately tracks RTT growth.
- The proposed method closely follows Global-1ms, confirming that fast-mode activation correctly adapts to increased RTT.

This result demonstrates that the proposed method scales correctly with real maritime communication delays, whereas Global-1s masks or distorts physical latency trends.

D. Packet Loss Sensitivity at Low RTT (10 ms)

Fig.6 examines packet loss rates from 0% to 20% under $RTT = 10$ ms. Global-1s again shows a fixed latency near 1 s, unable to capture retransmission effects or small delay increments. Global-1ms and the proposed method both exhibit a gradual increase in handshake duration as retransmissions occur. The proposed method remains nearly identical to Global-1ms, validating that retransmission timers inside the communication FMU trigger fast-mode execution correctly.

E. Packet Loss Sensitivity at High RTT (100 ms)

Fig.7 presents the same packet-loss experiment with $RTT = 100$ ms. Here, retransmission delays are significantly larger due to longer RTT and RTO intervals. The proposed method closely traces the Global-1ms trend, exhibiting linear growth in latency as the loss probability increases. Global-1s consistently overestimates latency and shows stepwise distortion due to 1-second quantization.

These results demonstrate that the proposed method remains robust under both loss-free and high-loss maritime network regimes, accurately reflecting transport-layer timing without excessive slow-FMU computation.

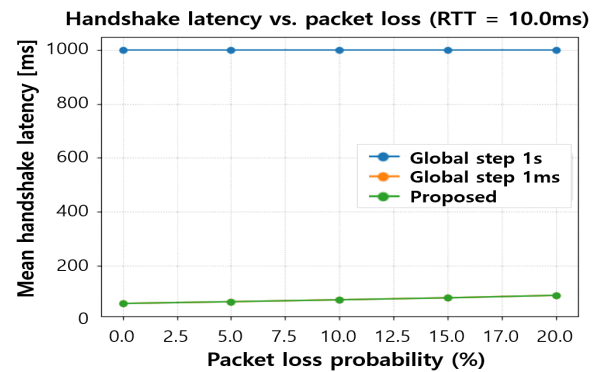


Fig. 6. Handshake Latency with RTT=10ms

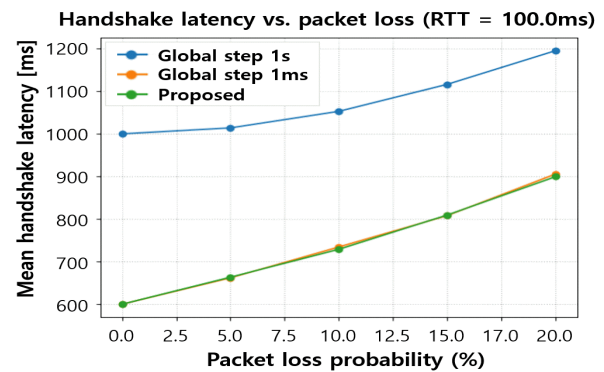


Fig. 7. Handshake Latency with RTT=100ms

F. Summary

Across all five experiments, the proposed method:

- preserves millisecond-level communication fidelity,
- avoids unnecessary slow-FMU invocations,
- scales correctly under varying RTT and packet-loss conditions,
- and eliminates quantization artifacts inherent in uniform large-step co-simulation.

This confirms that protocol-driven synchronization is essential for maritime scenarios where communication and physical dynamics span different time scales.

V. DISCUSSION

The experimental results across five scenarios demonstrate the importance of communication-aware synchronization in maritime co-simulation environments. Maritime communication links—such as VDES, Inmarsat-based satellite paths, ship-to-shore microwave relays, and coastal LTE/5G—typically exhibit higher latency, larger jitter, and non-negligible packet loss compared to terrestrial networks. Under these conditions, a uniform macro-step in FMI 2.0 co-simulation fails to represent protocol timing accurately: a large step size leads to severe quantization of handshake and retransmission events, while a small step size causes unnecessary computational load on slow FMUs such as ship dynamics.

The proposed method overcomes this limitation by making the communication FMU's internal protocol state a first-class input to the master algorithm. When the Communication FMU reports active protocol behavior (e.g., during TCP connection establishment, retransmissions, or command transfer), the master enters fast mode and advances only the fast cluster using the fine-grained step H_{fast} . When communication becomes idle, the master switches to slow mode with step H_{slow} , invoking both the communication-related FMUs and the ship-dynamics FMU D.

This design yields several benefits. First, handshake latency is no longer dominated by numerical quantization; the observed latencies almost match those of a global millisecond-level configuration even when RTT or packet loss varies. Second, because FMU D is evaluated only in slow mode, the number of its invocations stays close to that of a global 1 s configuration, avoiding thousands of redundant computations. Third, under conditions of high RTT or non-zero packet loss, the proposed method accurately reproduces timeout and retransmission effects without exaggeration or distortion.

These findings highlight the key insight of this work: *protocol-driven synchronization*, in which FMUs are scheduled based on their internal semantic activity rather than purely on numerical properties of the solvers, is essential for realistic and efficient modeling of coupled communication-control-dynamics systems in maritime digital twins. The approach is particularly valuable under the communication characteristics found in MASS (Maritime Autonomous Surface Ships) scenarios, where large RTTs, intermittent links, and multi-hop ship-port-cloud paths are common.

VI. CONCLUSION

This paper presented a protocol-aware dynamic synchronization method for multi-rate co-simulation in maritime systems. Traditional uniform-step execution cannot capture both the millisecond-scale timing of communication protocols and the second-scale evolution of physical dynamics. The proposed approach overcomes this issue by clustering fast and slow components, switching execution modes based on protocol-state information, and emulating event-driven behavior through zero-order hold and pseudo-clock mechanisms within an FMI-compatible workflow.

Across five experiment groups, the method preserved realistic TCP timing even under large RTT and packet-loss variations, while keeping slow-dynamics computations close to those of a coarse-step baseline. Compared to a global millisecond-level simulation, it avoids excessive evaluations of slow FMUs; compared to a global second-level approach, it eliminates quantization artifacts in handshake and retransmission timing. These results confirm that protocol-driven scheduling significantly enhances both fidelity and computational efficiency.

Looking forward, several research directions emerge:

- **FMI 3.0 Adoption:** Replacing pseudo-clock mechanisms with native clocked interfaces would improve abstraction and interoperability.
- **Adaptive Step Management:** Dynamically tuning H_{fast} , H_{slow} , and cluster membership based on network conditions or CPU load could improve efficiency.
- **Scalable Maritime Digital Twins:** Extending the approach to multi-ship, multi-port, and cloud-integrated scenarios would support large-scale validation of MASS operations.
- **Hardware-in-the-Loop Integration:** Incorporating real maritime communication devices (VDES, L-band, 5G) would support certification-oriented testing and hybrid simulation.

In summary, protocol-aware dynamic synchronization provides a practical and effective method to improve fidelity and efficiency in maritime co-simulation environments built around FMI-compatible tool chains such as OSP. This work lays the foundation for next-generation semantics-aware master algorithms for autonomous and remote maritime operations.

ACKNOWLEDGMENT

This research was supported by Korea Institute of Maritime Science & Technology Promotion(KIMST) funded by the Ministry of Oceans and Fisheries, Korea(RS-2022-KS221652, The Development of Simulation-based Evaluation Technology).

REFERENCES

- [1] *Functional Mock-up Interface for Model Exchange and Co-Simulation*, Modelica Association, Linköping, Sweden, 2014, specification document. [Online]. Available: <https://fmi-standard.org>
- [2] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: a survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–33, 2018.
- [3] International Maritime Organization, "Regulatory scoping exercise for the use of maritime autonomous surface ships (mass)," MSC.1/Circ.1604, 2021, maritime Safety Committee circular on MASS.
- [4] F. Lázaro, R. Raulefs, W. Wang, F. Clazzer, and S. Plass, "Vhf data exchange system (vdes): an enabling technology for maritime communications," *CEAS space Journal*, vol. 11, no. 1, pp. 55–63, 2019.
- [5] G. Maral, M. Bousquet, and Z. Sun, *Satellite communications systems: systems, techniques and technology*. John Wiley & Sons, 2020.
- [6] Open Simulation Platform, "Open simulation platform: towards a maritime ecosystem for efficient co-simulation," [urlhttps://opensimulationplatform.com/](https://opensimulationplatform.com/), 2022, Accessed: 2025-12-04.
- [7] I. Hafner and N. Popper, "An overview of the state of the art in co-simulation and related methods," *Simul. Notes Eur.*, vol. 31, no. 4, pp. 185–200, 2021.
- [8] A. A. van der Meer, P. Palensky, K. Heussen, D. M. Bondy, O. Gehrke, C. Steinbrink, M. Blanki, S. Lehnhoff, E. Widl, C. Moyo *et al.*, "Cyber-physical energy systems modeling, test specification, and co-simulation based testing," in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, 2017, pp. 1–9.
- [9] C. Gomes, T. Blochwitz, C. Bertsch, K. Wernersson, K. Schuch, O. Kotte, I. Zacharias, M. Blesken, T. Sommer, M. Najafi *et al.*, "The fmi 3.0 standard interface for clocked and scheduled simulations," in *Modelica Conferences*, 2021, pp. 27–36.
- [10] H. Georg, S. C. Müller, C. Rehtanz, and C. Wietfeld, "Analyzing cyber-physical energy systems: The inspire cosimulation of power and ict systems using hla," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2364–2373, 2014.
- [11] J. Kim and H. P. amd Woo-Sung Jung amd Dae Seung Yoo, "Study on co-simulation of heterogeneous simulators using functional mock-up interface and open-source toolchains," in *2025 16th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2025, pp. 843–847.