Liaison: an open-source tool for distributed co-simulations

Luis Sanchez-Heres[1] Fredrik Olsson[2] Jan Östh[3]

[1,2,3] Maritime department, RISE Research Institutes of Sweden, Sweden,

[1] luis.sanchez-heres@ri.se, [2] fredrik.x.olsson@ri.se

[3] jan.osth@ri.se

Abstract

This paper presents Liaison a new open-source tool designed to address challenges related to portability, security, and intellectual property when sharing Functional Mock-up Units. Built on the Functional Mock-up Interface 3.0 standard, Liaison has a server-client architecture that leverages Zenoh for communication. Zenoh is a novel pub/sub/query protocol that supports a variety of network topologies such as peer-to-peer, routed, mesh and brokered communication with minimal configuration, setting Liaison apart from similar tools. An example of a case that led the authors to develop this tool is presented.

Keywords: distributed co-simulations, maritime onboard systems

1 Introduction

The Functional Mock-up Interface (FMI) (Blochwitz et. al, 2011, Blochwitz et. al, 2012, Junghanns et. al, 2021) is a widely adopted standard in system simulation, providing a versatile and vendor-neutral framework for model exchange (ME) and co-simulations (CS) (Gomes et. al, 2018). At its core, FMI defines a set of conventions and interfaces that govern how simulation models interact with simulation environments, allowing for the encapsulation of models as Functional Mock-up Units (FMU). A FMU encapsulates a simulation model along with their associated parameters, inputs, and outputs, as compiled code black-boxes enabling the exchange and utilization across simulation tools that has implemented support for the FMI standard. For ME, the FMU does not include the model's solver, whereas for CS, the FMU also includes the solver of the model.

In practice, several factors can make sharing FMUs unfeasible. Such factors can include, but are not limited to:

- Portability issues (e.g. different operating systems or license requirements for execution of the FMU)
- Intellectual Property (IP) protection.
- Fear of unintended re-distribution of the FMU
- Fear of a FMU containing malicious code.

To overcome these obstacles, a distributed approach can be utilized where the FMU stays within the control and premises of the model owner but connects to a simulation environment over a network when desired. The connection can be made over a local area network or a wide area network such as the Internet.

Table 1 summarizes the issues solved by running distributed co-simulations locally, in a LAN, and in a WAN.

Table 1. Potential issues solved by running distributed co-simulations.

Issue	*Local*	*LAN*	*WAN*
FMU portability	x	x	x
Security	x	x	x
IP protection			x

FMU portability can be addressed by running distributed co-simulations locally, in a LAN, and in a WAN. In both LAN and WAN setups, a computer with a suitable operating system, environment, or hardware, can be used to serve the FMU, while another computer can be used to run the simulation. Meanwhile, security and IP protection can be address by running distributed co-simulations on a WAN across organisations. In this use case, one organisation creates a client FMU from a source FMU and delivers it to another organisation. This approach enables co-simulations without the transfer of intellectual property or risk of running malicious code.

Several tools for such distributed co-simulations have been developed. The most relevant for the work presented in this paper, is the FMU-Proxy (Hatledal et al., 2019) which is an open-source framework for distributed co-simulations supporting FMI 1.0 and 2.0. The original FMU is wrapped into a proxy FMU that is accessed through Remote Procedure Calls (RPCs). In this way, the platform becomes independent of both the FMU implementation language as well as the platform it runs on. FMU-Proxy relies on a combination of different

communication protocols such as HTTP, HTTP2, Websockets and TCP/IP.

In this paper we present Liaison (Liaison 2025), a new open-source tool for distributed co-simulations based on the FMI 3.0 standard and the Zenoh communication protocol (Corsaro et al., 2023). The decision to develop a new tool for distributed co-simulations was motivated by that existing tools at the time of project start did not support FMI 3.0, and that the adoption of Zenoh promised a simple and efficient communication framework (Palma, 2024) supporting also geographically distributed nodes deployed across wide area networks with minimal required network settings configuration (such as opening ports on host and client computers, etc). Zenoh runs on top of traditional communication protocols such as TCP/IP and UDP/IP and transparently bridge across different protocols. It is a pub/sub/query protocol and supports a variety of network topologies such as peer-to-peer, routed, mesh and brokered communication.

This introduction is followed by Section 2 where we present the high-level architecture of Liaison, network topologies and describes its implementation. In Section 3 we present an example case that led the authors to develop this tool (Software-in-the-Loop testing of sail control systems for wind assisted propulsion of ships (Laursen et al. 2023)). Section 4 presents some results connected to the computational overhead for the different use cases and Section 5 presents the outlook and future work.

2 Liaison

Figures 1 to 3 present the three basic configurations on which Liaison can be used. In all the figures, an "original" FMU is made available to a simulation software (co-simulation engine) with a Liaison FMU and a Liaison Server. A Liaison FMU is a client application that substitutes the original FMU and communicates with a Liaison Server. A Liaison Server is a server application that forwards the function calls made to the Liaison FMU by the simulation software to the original FMU.

Figure 1 and 2 present the configurations for running a distributed co-simulation locally and over a LAN. Both configurations are useful for addressing security and portability issues. For example, the FMU can be run inside of a sandbox/Docker environment or in another computer with the appropriate operating system or hardware resources. Meanwhile, Figure 3 corresponds to running a distributed co-simulation in a WAN. In addition to addressing portability and security issues, this configuration also enables running co-simulation without sharing IP across organizations, avoiding the need for non-disclosure agreements, software licenses, obfuscation, and similar practices.

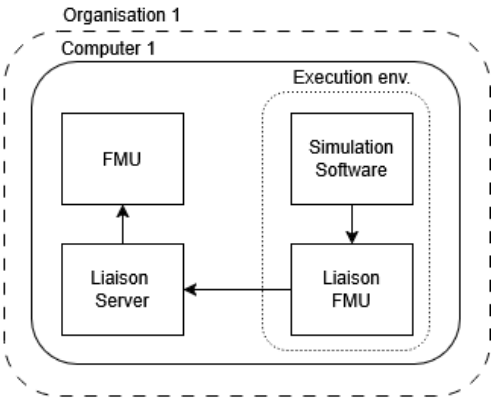

Figure 1. Configuration for running a distributed co-simulation locally with Liaison.

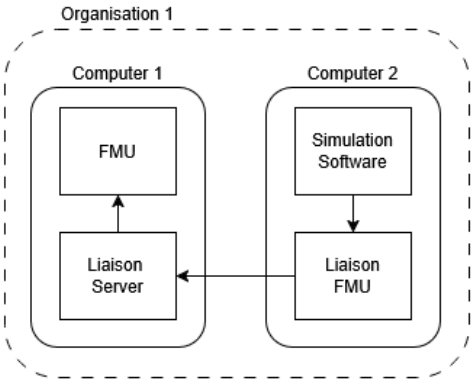

Figure 2. Configuration for running a distributed co-simulation over a LAN with Liaison.

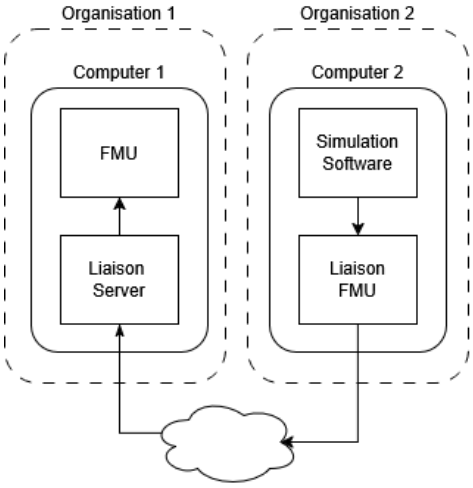

Figure 3. Configuration for running a distributed simulation over a WAN with Liaison.

2.1 Networking

Liaison uses Zenoh for communication. By default, Liaison operates using the "peer" mode in Zenoh which entails that:

- Every node discovers other nodes through a discovery mechanism.

DOI 10.3384/ecp218639

- Every node connects to and maintains a connection to all other nodes found through the discovery mechanism.

A schematic of this setup is illustrated in Figure 4. This setup is a convenient default as it allows for seamless, zero-configuration connections in a LAN.

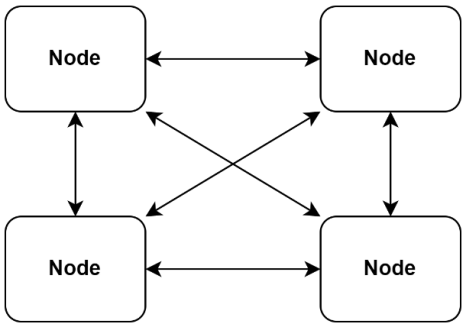

Figure 4. Schematic of 4 nodes connected in peer-to-peer mode. I.e. every node has a direct connection with every other node.

Further, Liaison allows for unimpeded configuration of the underlying Zenoh library. This allows for a range of different network topologies to be used, for details see (Corsaro et al., 2023). The most interesting one for the use of Liaison is the "client" mode which entails that:

- Every node connects to a well-known router on startup.
- The router ensures end-to-end connectivity in between all nodes connected to it.

A schematic of this setup is illustrated in Figure 5.

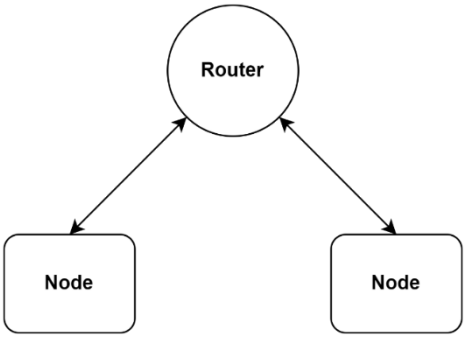

Figure 5. Schematic of two nodes connected in client mode via a (software) router. Each node maintains only a single connection to the router and the router ensures end-to-end connectivity in between nodes.

Although the "client" mode of communication in Zenoh requires every node to make an initial connection to a well-known router, it does not require every node to know the connectivity details (such as IP number and port) of all the other nodes in the network. Effectively, this allows for nodes in different LANs, possibly geographically distributed, to interconnect via a router deployed on a well-known address accessible over a WAN (such as the internet) without the need of knowing the connectivity details of each other. A schematic of this setup is outlined in Figure 6.

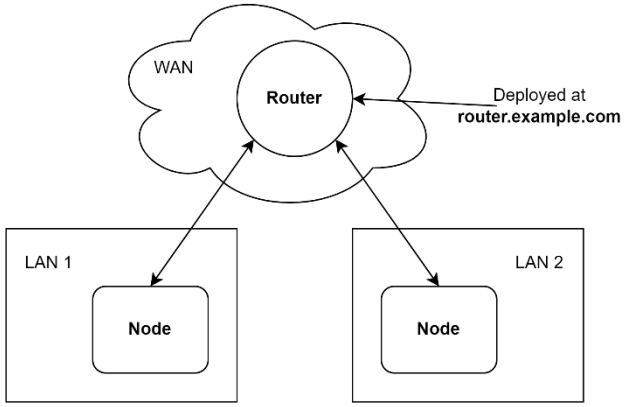

Figure 6. Schematic of 2 nodes connecting across a WAN from two separate LANs in client mode via a (software) router deployed on a well-known address.

Further, the underlying Zenoh library may be configured for secured communication channels, regardless of network topology, using either client/server TLS or mutual TLS.

2.2 Implementation

Liaison is an application written in C++ and compiled for Windows and Linux. It is released as open source under the Apache 2.0 license and is available for download at http://github.com/rise-maritime/liaison. While it depends on standard system libraries, third-party dependencies are bundled with the release, requiring no additional installation.

Liaison consists of a platform specific executable and a set of dynamic libraries that encapsulate the client logic for different platforms. The executable performs two main functions: 1) making a „Liaison FMU" from a standard FMU, and 2) serving an FMU. When making a Liaison FMU, the executable extracts the model description XML file from the source FMU, renames the dynamic libraries according to the model's name, and re-packages it all. In turn, when serving an FMU, the executable loads the source FMU and forwards to it all the incoming queries from the Liaison FMU as seen in Figures 1 to 3.

At the time of writing, only a subset of the FMI 3.0 standard is implemented. For an up to date list of the implemented functions the readers are referred to the project's online repository (Liaison, 2025).

3 Example: Software-In-The-Loop (SIL) for control of a Wind-Assisted Propulsion System

Consider the following made-up example: A ship owner is interested in retrofitting one of its vessels with a wind-assisted propulsion system (Laursen et al. 2023) and is particularly interested in rigid wings as suggested by a potential provider. But, prior to signing a deal, the shipowner wants to ensure the interoperability of the proposed wing sail system (and its controller) with the existing autopilot used on the ship. For that purpose, the ship owner demands a set of co-simulations to be performed to verify the interoperability in a Software-in-the-Loop (SIL) setup.

In this paper we will use the example above to exemplify the usefulness of Liaison in overcoming technical as well as organizational obstacles in sharing software, especially control systems, in the design process of ships. Further, we will use the same setup to investigate the performance penalty introduced by Liaison through different scenarios.

3.1 Simulation setup

Figure 7 presents the simulation setup with the following components:

- **Vessel FMU:** an FMU that encapsulates a hydrodynamic model of a cargo vessel as well as an aerodynamic model of a set of rigid wings mounted on said cargo vessel. The models used in this FMU are built using the RISE in-house ship dynamics solver SEAMAN (Ottosson and Byström, 1991).
- **Controller FMU**: an FMU that encapsulates the controller logic for the wing controller system.
- **Co-simulation engine:** A co-simulation engine that is responsible for stepping the co-simulation forward and handling connections between the two FMUs. The co-simulation engine used for these simulations is Ecos, for details see (Ecos 2025).

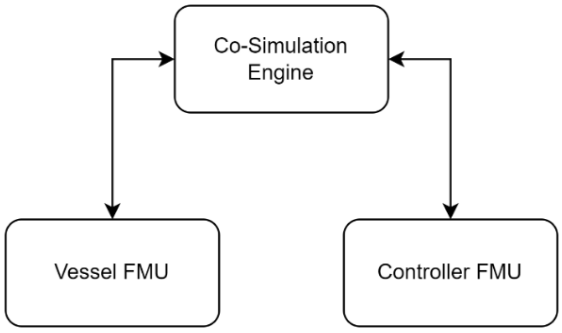

Figure 7. Simulation setup for the example.

The FMUs used in the simulation setup are deliberately kept at minimum level of complexity, using only the bare minimum of the FMI 3.0 specification, to ensure fair comparisons focusing on the performance penalty introduced by Liaison.

3.2 Scenarios

The simulated setup is used in the following scenarios:

- *Reference scenario:* the FMUs are loaded directly by the co-simulation engine, i.e. Liaison is not used.
- *Local scenario:* the FMUs and the co-simulation engine runs on the same physical computer, but Liaison is used as the means of communication.
- *LAN scenario*: the FMUs and the co-simulation engine run on different physical computers (3 computers in total), but in the same LAN.
- *WAN scenario:* the FMUs and the co-simulation engine all run on different physical computers (3 computers in total) on different LANs interconnected via a WAN (the Internet). The communication channels are secured using mTLS.

To investigate the performance penalty of using Liaison, each of the scenarios are run for 600 seconds with three different step sizes: 0.1, 1.0, and 10.0 seconds. To account for variations in execution time, each scenario and step size combination is simulated 10 times, and the average time is recorded.

4 Results

Figure 8 presents the performance penalty associated with using Liaison across various configurations and communication step sizes.

For a communication step size of 0.1 seconds:

- The reference scenario has an average execution time of about 25 seconds.
- The local scenario has an average execution time of 1.4 times that of the reference scenario.
- The LAN configuration scenario has an average execution time that is more than double (2.4 times) that of the reference scenario.
- The WAN scenario has a significantly higher average execution time, 7.6 times that of the reference scenario.

For a communication step size of 1.0 second:

- The reference, local, and WAN scenarios have similar execution times at about 25 seconds.
- The WAN scenario has a higher average execution time, about 1.7 times that of the other scenarios.

For a communication step size of 10.0 seconds:

- All cases exhibit similar average execution times, approximately 25 seconds.

DOI 10.3384/ecp218639

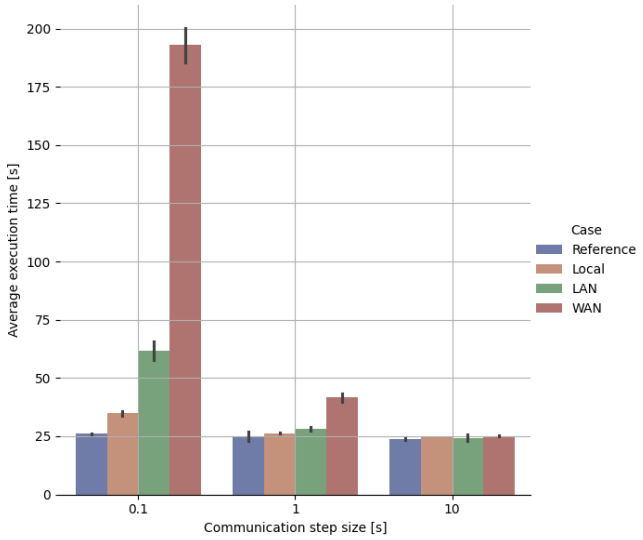

Figure 8. Performance results from different co-simulation scenarios and configurations.

These results indicate that the communication step size has a substantial impact on execution time, particularly in the WAN scenario. The performance is notably better in scenarios with larger communication step sizes and when the components are either on the same physical computer or within the same LAN.

5 Conclusion and future work

Liaison is a general open-source tool for distributed co-simulations and model exchange based on FMI 3.0. The workflow when using Liaison starts with an existing FMU. In the first step, Liaison is used to produce a "Liaison FMU", that can be included in any simulation tool that supports FMI 3.0. For the simulation tool, the Liaison FMU appears as a standard FMU. The Liaison FMU acts as a client and includes all information it needs to communicate with a Liaison server over LAN or WAN (such as the Internet). The communication protocol in Liaison is the Zenoh protocol, which ensures a simple communication configuration by the user and versatile network topologies.

Results presented in this paper shows that running simulations in a distributed fashions with Liaison can have a negative impact on simulation time due to communication events between the distributed Liaison FMUs that are part of the distributed co-simulation. However, this performance drawback can be mitigated by selecting appropriate communication step sizes.

Future work will focus on achieving full compatibility with FMI 3.0 and include support for FMI 2.0, as well as testing Liaison in various simulation software environments. Contributions are welcome!

Acknowledgements

This work is part of the project "Virtual Wind Ship", carried out within the Swedish Transport Administration's industry program Sustainable Shipping, operated by Lighthouse".

References

Blochwitz, Torsten; Martin Otter; Martin Arnold, et al. (2011-03). "The functional mockup interface for tool independent exchange of simulation models." In: *Proceedings of the 8th International Modelica Conference.* Linköping University Press, pp. 105-114.

Blochwitz, Torsten; Martin Otter; Johan R. Åkesson; et al. (2012-09). "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models". In: *Proceedings of the 9th International Modelica Conference.* The Modelica Association, pp. 173-184.

Corsaro, Angelo; Luca Cominardi; Olivier Hecart; et al. (2023-09) "Zenoh: Unifying communication, storage and computation from the cloud to the microcontroller." In*: 26th Euromicro Conference on Digital System Design (DSD*). IEEE pp. 422-428.

Ecos (2025). Ecos co-simulation platform. Code repository. URL: *https://github.com/Ecos-platform/ecos (visited on 2025-07-30)*

Gomes, Cláudio; Casper Thule; David Broman; et al. (2018). "Co-simulation: a survey". In: *ACM Computing Surveys (CSUR)* 51(3):49, pp. 1-33.

Hatledal, Lars; Ivar Arne Styve; Geir Hovland and Houxiang Zhang (2019). "A language and platform independent co-simulation framework based on the functional mock-up interface". IEEE Access, Vol. 7 pp. 109328-109339. DOI: 10.1109/ACCESS.2019.2933275

Giese, Holger Giese; Dominique Blouin; Rima Al-Ali; et al. (2021). "Chapter 4 - An ontology for multi-paradigm modelling." Multi-Paradigm Modelling Approaches for Cyber-Physical Systems, Elsevier. ISBN; 978-0-12-819105-7

Junghanns, Andreas; Cláudio Gomes; Christian Schulze; et al. (2021-09). "The functional mock-up interface 3.0-new features enabling new applications." In: *Proceedings of the 14th Modelica Conference*, Linköping University Electronic Press, pp. 17–26. DOI: 10.3384/ecp2118117

Kolodziejski, Michal and Marcin Sosnows;ki (2025). "Review of Wind-Assisted Propulsion Systems in

Maritime Transport". Energies, 18(4):897. DOI: 10.3390/en18040897

Laursen, Rasmus; Hitesh Patel; Dimitra Sofiadi; et al. (2023). "Potential of Wind-Assisted Propulsion for Shipping." Tech. rep. *European Maritime Safety Agency (EMSA Report EMSA/OP/43/2020),* pp. 1-271

Liaison (2025). Liaison co-simulation tool. Code repository. URL: *https://github.com/RISE-Maritime/liaison (visited on 2025-07-30).*

Ottosson, Peter and Lars Byström (1991). "Simulation of the Dynamics of a Ship Manoeuvring in Waves". Transactions of the Society of Naval Architects and Marine Engineers (Trans. SNAME) 99, pp. 281–298

Palma, Steven (2024), "Transforming the robotics industry: A sustainable communication protocol". Control Engineering. URL: https://www.controleng.com/transforming-the-robotics-industry-a-sustainable-communication-protocol (visited on 2025-07-30).