FMI-BASED DISTRIBUTED CO-SIMULATION WITH ENHANCED SECURITY AND INTELLECTUAL PROPERTY SAFEGUARDS

Santiago Gil[a], Ecem E. Baş[b], Christian D. Jensen[a], Sebastian Engelsgaard[c], Giuseppe Abbiati[a], and Cláudio Gomes[a]

[a]Aarhus University, Aarhus, Denmark
[b]R&D Test Systems, Hinnerup, Denmark
[c]Lorc, Munkebo, Denmark

ABSTRACT

Distributed co-simulation plays a key role in enabling collaborative modeling and simulation by different stakeholders while protecting their Intellectual Property (IP). Although IP protection is provided implicitly by co-simulation, there is no consensus in the guidelines to conduct distributed co-simulation of continuous-time or hybrid systems with no exposure to potential hacking attacks. We propose an approach for distributed co-simulation on top of UniFMU with enhanced cybersecurity and IP protection mechanisms, ensuring that the connection is initiated by the client and the models and binaries live on trusted platforms. We showcase the functionality of this approach using two co-simulation demos in four different network settings and analyze the trade-off between IP-protected distribution and performance efficiency in these settings.

Keywords: Distributed co-simulation, IP protection, collaborative modeling and simulation.

1 INTRODUCTION

The increasing complexity of modern engineering systems has driven the need for collaborative modeling and simulation approaches that enable stakeholders to work together seamlessly. Distributed co-simulation has emerged to ease the integration and co-development of heterogeneous models and simulations, allowing organizations to contribute their expertise while retaining control over their intellectual property (IP). However, its adoption is hindered by concerns over cybersecurity and IP protection, especially when a company's model must run on *untrusted platforms* (computers outside the control of the company's IT department) with standalone software. This problem is recognized in [1], which presents an approach for distributed co-simulation with secure communication between master algorithm and models. However, connections that are initiated from the master (untrusted platform) to the model (trusted platform) pose a security risk that IT departments are often unwilling to take since it requires the trusted platform to have open ports.

Similarly, code executed on untrusted platforms can be extracted or reverse-engineered from memory, as attackers can use debugging tools, memory dumps, or other techniques to analyze and retrieve sensitive data. This poses a substantial risk to proprietary information, as critical algorithms, configuration parameters, or proprietary logic embedded in the code can be exposed, replicated, or misused by unauthorized parties.

This paper introduces a tool (as an extension of the UniFMU tool [2]) and a methodology for IP-protected distributed co-simulation for collaborative engineering across companies on top of the Functional Mock-up Interface (FMI) standard. Unlike conventional distributed co-simulation tools, which often require opening network ports or deploying models on potentially untrusted platforms (see Section 2), our approach

Proc. of the 2025 Annual Modeling and Simulation Conference (ANNSIM'25)
May 26-29, 2025, Universidad Complutense de Madrid, Madrid, Spain
J.L. Risco-Martín, G. Rabadi, D. Cetinkaya, R. Cárdenas, S. Ferrero-Losada, and A. Bany Abdelnabi, eds.

provides a convenient deployment architecture where connections are initiated from trusted platforms hosting the models, eliminating the need for exposed ports and thereby reducing susceptibility to cyberattacks. Moreover, by eliminating the need for models to run on untrusted platforms and keeping them on trusted machines on the client side, our tool enables secure collaboration, particularly in scenarios involving multiple organizations with stringent IP protection requirements, while keeping efficient and adaptable simulation workflows by replacing proxy models with their real counterparts as needed.

2 STATE OF THE ART

Co-simulation is a technology that extends the capabilities of simulation, which has been used to answer questions of the type *what-if*, by enabling simulations via the composition of simulation units [3].

Existing standards for co-simulation include FMI, which defines the guidelines for Model Exchange and co-simulation for continuous-time co-simulations; the IEEE 1516-2010 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), which defines the guidelines for distributed (co-)simulations (*Federations*) in discrete-event co-simulations via a common Real-Time Infrastructure (RTI); the Distributed Co-simulation Protocol (DCP), which provides the guidelines for standardized interoperability on distributed hardware platforms extending from FMI; and the IEEE 1730-2022 Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP), which defines a high-level framework for integrating distributed simulation of lower-level systems. In this work, we work with the FMI standard.

Table 1: Summary of approaches in the state of the art.

Contribution	Tool	Communication layer	Standard
Norling et al. [1]	TLM for co-simulation	Stand-alone over TCP	—
Falcone & Garro [4]	HLA Development Kit	HLA Pub/Sub API	FMI & HLA
Skjong et al. & Sadjina et al. [5, 6]	Coral (now called *The Open Simulation Platform*)	Pub/Sub by slave providers	FMI
Cakmak et al. [7]	—	Models (FMUs) embedded with TCP-based REST APIs	FMI
Schiera et al. [8]	Mosaik	Master/Slave over TCP (Mosaik Simulator API)	FMI
Meyer et al. [9]	xMOD	DCP-based over UDP	DCP
Zhao et al. [10]	LICPIE	Stand-alone over TCP & UDP	—
Krammer et al. [11]	DCPLib	DCP-based over UDP	DCP & DSEEP
Junior et al. [12]	Ptolemy, CertiHLA, & PyHLA	HLA Pub/Sub API	HLA
Segura et al. [13]	Simulink library	DCP-based	DCP
Reiher & Hahn [14]	Portico	HLA Pub/Sub API	HLA
Hong et al. [15]	FMU SDK (now managed by the Modelica Association)	Data Distribution Service Pub/Sub	FMI
Dad et al. [16]	DACCOSIM	ZeroMQ middleware	FMI
Rautenberg et al. [17]	—	DCP-based over UDP	DCP
Mao et al. [18]	DRLFluent	CORBA	—
Mehlmann et al. [19]	VILLAS Framework	MQTT, Kafka, FIWARE, WebRTC, RTP, UDP, and TCP	—

Existing works in the literature tackle the need for distributed co-simulation, especially the recent DCP protocol (for continuous-time simulation) and the HLA standard (for discrete-event simulation). However, there is yet no optimal solution (and guidelines) to conduct distributed co-simulation of continuous-time or hybrid systems which guarantees no exposure to cyber attacks of the machine where the IP-protected models run. Table 1 shows a summary of contributions in the literature that have addressed techniques to implement distributed co-simulation using different tools, communication methods, and standards. Of the 16 entries, six work with FMI, three with HLA, four with DCP, and the rest are not specified. From these, we can also identify that those following HLA and DCP are more standardized at the communication layer, whereas those following either FMI or none are more diverse. Although all the approaches provide IP protection implicitly, extra cybersecurity mechanisms to avoid vulnerability of the machine where the IP is and of the models and binaries include: [4, 19] ensure that the models and binaries do not need to be shared with third parties, [10, 18] provide session control for simulation units, and [1] has encryption in the messaging between the master algorithm and the simulation units, but the connection is initiated from the

master, i.e., an untrusted platform. Our approach also ensures that the models and binaries do not need to be shared with third parties, making the IP hermetic to hostile platforms or users; additionally, the connection is started from the client-side and, therefore, there is no need for open ports or firewall control.

3 APPROACH

We propose a technique to carry out distributed co-simulation based on the FMI standard. To do so, we build on top of the existing UniFMU [2], which already provides the capabilities to split the FMU binary from the model using a Pub/Sub connection over ZeroMQ. Following the same concept, we split the FMU binary and the model using different processes, which may be executed in different network clients, i.e., one process for the FMU binary and one process for the model execution.

In this approach, the FMU works as a proxy (without any model), which waits for a connection from the model, through a ZeroMQ connection. The proxy and the model exchange data in the form of one-to-one acknowledged Protobuf messages following FMI over ZeroMQ, where participants can only access data streams of their own models. The proxy forwards the messages from the co-simulation master algorithm to the model and vice-versa, as shown in Figure 1. The master algorithm, on the other hand, has access to data streams generated from inputs and outputs of the orchestrated proxies and other eventual FMUs.

When the master algorithm calls the `instantiate` function on a proxy-model pair, this enters a blocking state, waiting for the client, i.e., the model, to connect to the ZeroMQ broker created by the FMU proxy. When the model connects to the ZeroMQ broker, the co-simulation starts until termination, releasing the resources of the two processes, that is, proxy and model. This occurs for each proxy-model pair in the co-simulation. The master algorithm can combine FMUs whether they are proxies or not.

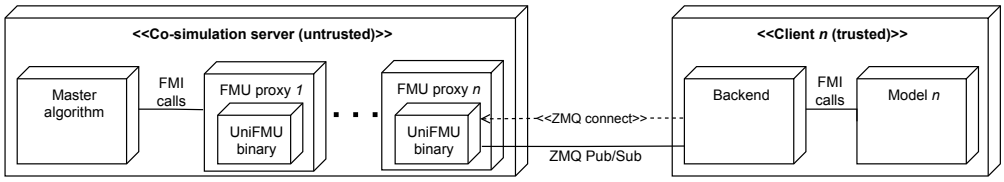

Figure 1: Deployment diagram of distributed co-simulation with UniFMU.

It is worth mentioning that the connection is established by the client (the trusted platform), avoiding the need for open ports and firewall control for the connection on the client side, i.e., where the IP lives. Thus, the open ports are only needed on the proxy side (the untrusted platform), where the collaborative co-simulation is executed, without access to the IP. At connection time, authentication can also be provided. This approach can also run distributed co-simulations with existing FMUs, where the box *Model n* in Figure 1 is replaced by the existing FMU, and the *FMI calls* are executed on it using the FMPy library [20].

4 DEMONSTRATION AND LIMITATIONS

To demonstrate the use of this approach, we provide two publically available demos on GitHub [21]. In the first demo (**Demo 1**), we use the default template created by UniFMU, which provides a pair proxy-model with a set of variables to perform simple calculations of the type sum, rest, string concatenation, and boolean operations; all these operations are executed on demand by the master algorithm, in this case, using FMPy. In the second demo (**Demo 2**), we use a co-simulation with three FMUs that replicate a test bench for a 16 MW test facility of wind turbine nacelles. The three FMUs represent the test bench controller, test bench motor, and the drive train generator of the device under test. This demo uses Maestro [22].

For **Demo 1**, we initialize a co-simulation of the proxy-model pair with FMPy, and iterate through one thousand steps with step size $\Delta_t = 0.01$ s and simple operations, as described in this setting, more precisely, we run one thousand times a set of `setReal`, `doStep`, and `getReal` calls. For **Demo 2**, we execute a co-simulation with three proxy-model pairs for a total of 50.0 s of co-simulation time with a step size of $\Delta_t = 0.1$ s. Once the third FMU pair has established connection, the master algorithm proceeds with the co-simulation execution for 50.0 s given the co-simulation scenario. Both demos are executed in real-time and non-real-time modes. For the former, we expect that the average step time (AS_t) is $AS_t \simeq \Delta_t$; for the latter (normal simulation time), AS_t depends on the network and computation capabilities, usually $AS_t < \Delta_t$.

The main limitation is the performance degradation, due to network communication. To understand the trade-off between IP protection and *Performance Efficiency*, we follow the guidelines of the ISO/IEC 25010 for systems and software quality requirements and evaluation. For this, we measure the time behavior and resource utilization of the demos in four network settings. Settings **1** and **2** are for collaborative co-simulations where stakeholders can gather physically together, whereas settings **3** and **4** can be used for geographically distributed co-simulations, which may compromise security and performance. Both demos were tested in the four settings, providing the same co-simulation results with different computation times.

Wireless Local Area Network (WLAN) (Setting 1). In this setting, we distribute proxy and model processes on two different machines with a private connection over the WLAN using a domestic modem.
Switched LAN (Setting 2). In this setting, we use an industrial switch (HP Aruba 2930F) to distribute the two processes as in Setting 1, but this one uses a wired network on the data-link layer instead.
Public server from a non-controlled private network (Setting 3). In this setting, we distribute the processes on two different machines, where the proxies are hosted on a public server (an AWS EC2 instance), and the models are hosted on a private machine running on a non-controlled network.
Public server from an IT-controlled network (Setting 4). In this setting, we use the same settings as in **Setting 3**, but instead, the private machine runs on an IT-controlled network with more strict policies.

The performance efficiency results for time behavior of this approach are collected in Table 2. Notice that the resource utilization for **Demo 1** is minimal since the computation accounts for the sum of two real numbers and abuses the network resources, whereas the resource utilization for **Demo 2** is moderate, with the execution of the dynamics of a coupled system. In particular, for the last row (**Demo 2** in real-time mode), we clearly see that Settings **3** and **4** cannot perform in real-time as expected, exceeding the 50.0s of computation because every step takes longer than required ($AS_t > \Delta_t$), opposite to Settings **1** and **2**. With Table 2, we confirm there is a trade-off between hard real-time requirements and (geographical) distribution of IP-protected co-simulation due to incremental network delays. If hard or near-hard real-time is needed, following **Setting 2** is the recommended option, which is the fastest yet secure setup.

5 CONCLUDING REMARKS

In this work, we proposed an approach for distributed co-simulation following the FMI standard upon UniFMU. The demonstration in Section 4 shows that this approach for distributed co-simulation is useful when models need to be hidden from the co-simulation master algorithm, i.e., models are protected from running on external platforms, ensuring IP protection and integrity. This is also beneficial since the client hosts the model, avoiding dependen-

Table 2: Measures of performance for time behavior (all measures are in seconds).

Demo	Real-time mode	Setting 1	Setting 2	Setting 3	Setting 4
Demo 1	No	13.8128	1.4747	52.4733	63.6923
Demo 1	Yes	27.2524	16.6968	61.7995	74.5381
Demo 2	No	20.8260	3.2366	79.4714	93.6664
Demo 2	Yes	50.2971	50.1930	88.9710	93.8596

cies which may not be contained in the FMUs on the co-simulation server, and decides when to start the connection without the need for open ports or firewall control.

Although the use of strong authentication mechanisms and containerization can improve the security of tools that assume incoming connections on trusted platforms, in practice, engineers participating in these simulations do not necessarily have the required background or permissions to handle this. In our experience with IT departments, changing firewall settings to allow such connections is a cumbersome process.

Additionally, there are limitations for hard real-time applications with small step sizes due to network delays, especially if hardware-in-the-loop is involved. In such scenarios, it is suggested to work on a switched LAN with dedicated equipment or using other alternatives, such as DCP co-simulation with operation mode for hard or soft real-time. Our approach has still limited support for the FMI 3.0 version, which is a work in progress. In future work, we plan to improve the authentication and security mechanisms for data confidentiality and integrity of this approach and evaluate the performance of using Virtual LANs.

ACKNOWLEDGMENTS

This work has been funded and supported by the DIGIT-Bench project (case no. 640222-497272), funded by the Energy Technology Development and Demonstration Programme (EUDP). The authors are grateful to the DIGIT-Bench project partners and the reviewers for the discussions that contributed to this paper.

REFERENCES

[1] K. Norling, D. Broman, P. Fritzson, A. Siemers, and D. Fritzson, "Secure distributed co-simulation over wide area networks," in *Proc. of the 48th SIMS conf.* Citeseer, 2007, pp. 14–23.

[2] C. M. Legaard, D. Tola, T. Schranz, H. D. Macedo, and P. G. Larsen, "A universal mechanism for implementing functional mock-up units," in *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, ser. SIMULTECH, 2021, pp. 121–129.

[3] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Computing Surveys*, vol. 51, no. 3, 2018.

[4] A. Falcone and A. Garro, "Distributed Co-Simulation of Complex Engineered Systems by Combining the High Level Architecture and Functional Mock-up Interface," *Simulation Modelling Practice and Theory*, vol. 97, no. August, p. 101967, 2019.

[5] S. Skjong, M. Rindarøy, L. T. Kyllingstad, V. Æsøy, and E. Pedersen, "Virtual prototyping of maritime systems and operations: applications of distributed co-simulations," *Journal of Marine Science and Technology (Japan)*, vol. 23, no. 4, pp. 835–853, 2018.

[6] S. Sadjina, L. T. Kyllingstad, M. Rindarøy, S. Skjong, V. Æsøy, and E. Pedersen, "Distributed Co-simulation of Maritime Systems and Operations," *Journal of Offshore Mechanics and Arctic Engineering*, vol. 141, no. 1, p. 011302, 09 2018.

[7] H. Çakmak, A. Erdmann, M. Kyesswa, U. Kühnapfel, and V. Hagenmeyer, "A new distributed co-simulation architecture for multi-physics based energy systems integration: Analysis of multimodal energy systems," *At-Automatisierungstechnik*, vol. 67, no. 11, pp. 972–983, 2019.

[8] D. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. Bompard, E. Patti, E. Macii, and L. Bottaccioli, "A Distributed Multimodel Platform to Cosimulate Multienergy Systems in Smart Buildings," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4428–4440, 2021.

[9] M. A. Meyer, L. Sauter, C. Granrath, H. Hadj-Amor, and J. Andert, "Simulator Coupled with Distributed Co-Simulation Protocol for Automated Driving Tests," *Automotive Innovation*, vol. 4, no. 4, pp. 373–389, 2021.

[10] L. Zhao, M. Ni, H. Tong, and Y. Li, "Design and application of distributed co-simulation platform for cyber physical power system based on the concepts of software bus and middleware," *IET Cyber-Physical Systems: Theory and Applications*, vol. 5, no. 1, pp. 71–79, 2020.

[11] M. Krammer, C. Schiffer, and M. Benedikt, "ProMECoS: A process model for efficient standard-driven distributed co-simulation," *Electronics*, vol. 10, no. 5, pp. 1–26, 2021.

[12] J. C. V. Junior, A. V. Brito, L. F. S. Costa, T. P. Nascimento, and E. U. K. Melcher, "Testing real-time embedded systems using high level architecture," *Design Automation for Embedded Systems*, vol. 20, no. 4, pp. 289–309, 2016.

[13] M. Segura, T. Poggi, and R. Barcena, "A Generic Interface for x-in-the-Loop Simulations Based on Distributed Co-Simulation Protocol," *IEEE Access*, vol. 11, no. January, pp. 5578–5595, 2023.

[14] D. Reiher and A. Hahn, "Ad hoc HLA simulation model derived from a model-based traffic scenario," *Simulation*, vol. 99, no. 8, pp. 859–882, 2023.

[15] S. Hong, D. Lim, I. Joe, and W. Kim, "F-DCS: Fmi-based distributed CPS simulation framework with a redundancy reduction algorithm," *Sensors (Switzerland)*, vol. 20, no. 1, 2020.

[16] C. Dad, J. P. Tavella, and S. Vialle, "Synthesis and feedback on the distribution and parallelization of FMI-CS-based co-simulations with the DACCOSIM platform," *Parallel Computing*, vol. 106, 2021.

[17] P. Rautenberg, P. Weber, J. P. Degel, S. Hähnlein, F. Gauterin, T. Koch, M. Doppelbauer, and M. Gohl, "Electrified Powertrain Development: Distributed Co-Simulation Protocol Extension for Coupled Test Bench Operations," *Applied Sciences (Switzerland)*, vol. 13, no. 4, 2023.

[18] Y. Mao, S. Zhong, and H. Yin, "DRLFluent: A distributed co-simulation framework coupling deep reinforcement learning with Ansys-Fluent on high-performance computing systems," *Journal of Computational Science*, vol. 74, no. October, p. 102171, 2023.

[19] G. Mehlmann, U. Kühnapfel, F. Wege, A. Winkens, C. Scheibe, S. Vogel, P. Noglik, M. Gratza, J. Richter, M. Weber, I. Burlakin, A. Kuri, T. Wagner, S. Hubschneider, R. Poppenborg, T. Heins, K. Förderer, A. Ulbig, A. Monti, V. Hagenmeyer, and M. Luther, "The Kopernikus ENSURE Co-Demonstration Platform," *IEEE Open J. of Power Electronics*, vol. 4, pp. 987–1002, 2023.

[20] "FMPy," https://github.com/CATIA-Systems/FMPy, [Online; acc. 02-April-2025].

[21] "UniFMU demos," https://tinyurl.com/nha6kcfm, [Online; acc. 02-April-2025].

[22] "Maestro 2," https://github.com/INTO-CPS-Association/maestro, [Online; acc. 02-April-2025].

AUTHOR BIOGRAPHIES

SANTIAGO GIL is a post-doc at Aarhus University, where he completed his PhD in 2024. His research interests include Digital Twins, co-simulation, and Internet of Things. Email address: sgil@ece.au.dk.

ELIF E. BAŞ holds a PhD in Structural Engineering, with a research focus on hybrid testing. She develops innovative experimental facilities for wind power at R&D Test Systems. Email address: eeb@rdas.dk

CHRISTIAN D. JENSEN is a Professor of Cybersecurity at Aarhus University. He holds a PhD from Université Joseph Fourier. He focuses on trust-based methods and technologies to secure collaboration among entities in open distributed systems. Email address: cdj@ece.au.dk

SEBASTIAN ENGELSGAARD is a Test System Developer at LORC. He specializes in developing and optimizing test systems for offshore renewable energy technologies. Email address: se@lorc.dk

GIUSEPPE ABBIATI is an Associate Professor and Head of section at Aarhus University. He leads research on developing DTs for the mechanical and construction industries. Email address: abbiati@cae.au.dk

CLÁUDIO GOMES is an Assistant Professor at Aarhus University. He holds a PhD from the University of Antwerp. His research focuses co-simulation and Digital Twins. Email address: claudio.gomes@ece.au.dk.