Comput Sci Res Dev
DOI 10.1007/s00450-017-0379-y

energieinformatik2017
The Swiss-German-Austrian Research Summit on Energy Informatics

SPECIAL ISSUE PAPER

Smart grid co-simulation with MOSAIK and HLA: a comparison study

C. Steinbrink[1] · **A. A. van der Meer**[2] · **M. Cvetkovic**[2] · **D. Babazadeh**[1] · **S. Rohjans**[3] · **P. Palensky**[2] · **S. Lehnhoff**[1]

Abstract Evaluating new technological developments for energy systems is becoming more and more complex. The overall application environment is a continuously growing and interconnected cyber-physical system so that analytical assessment is practically impossible to realize. Consequently, new solutions must be evaluated in simulation studies. Due to the interdisciplinarity of the simulation scenarios, various heterogeneous tools must be connected. This approach is known as co-simulation. During the last years, different approaches have been developed or adapted for applications in energy systems. In this paper, two co-simulation approaches are compared that follow generic, versatile concepts. The tool MOSAIK, which has been explicitly developed for the purpose of co-simulation in complex energy systems, is compared to the High Level Architecture (HLA), which possesses a domain-independent scope but is often employed in the energy domain. The comparison is twofold, considering the tools' conceptual architectures as well as results from the simulation of representative test cases. It suggests that MOSAIK may be the better choice for entry-level, prototypical co-simulation while HLA is more suited for complex and extensive studies.

Keywords Co-simulation · MOSAIK · HLA · Cyber-physical energy systems

✉ C. Steinbrink
steinbrink@offis.de

A. A. van der Meer
a.a.vandermeer@tudelft.nl

M. Cvetkovic
m.cvetkovic@tudelft.nl

D. Babazadeh
babazadeh@offis.de

S. Rohjans
sebastian.rohjans@haw-hamburg.de

P. Palensky
p.palensky@tudelft.nl

S. Lehnhoff
lehnhoff@offis.de

[1] OFFIS – Institute for Information Technology, Oldenburg, Germany

[2] Delft University of Technology, Delft, The Netherlands

[3] Hamburg University of Applied Sciences, Hamburg, Germany

1 Introduction

The evaluation of new planning, operation and control designs in the energy domain requires thorough analysis. In the recent years, the overall character of the domain has turned to the field of cyber-physical energy systems (CPES) with smart grids as one of its most prominent concepts. This development leads to complex, interdisciplinary setups that are infeasible to handle with purely analytical evaluation. Since testing new approaches in laboratories or in the field is too inflexible and expensive for early development stages, it is an established procedure to conduct co-simulation beforehand. This technique is defined as the coordinated execution of two or more simulation models that differ in their representation as well as in their runtime environment [1].

The first co-simulation implementations have mostly been focused on analysis of the interaction between power system and communication network simulation models [2–5]. However, besides this direct coupling of two tools, more generic co-simulation approaches have been developed during the last years. They are called *co-simulation frameworks* because one of their integral parts is a middleware that is responsible for data exchange and temporal synchronization of sev-

Published online: 04 September 2017

eral simulation models. Many prominent examples for such frameworks are either based on the PTOLEMY II software [6] (e. g. the Building Controls Virtual Test Bed [7]) or on various implementations of the High-Level Architecture standard (HLA) for distributed simulation systems [8] (e. g. C2WT-TE [9]). A rather new example for a CPES co-simulation framework is the tool MOSAIK that has been developed at the Oldenburg Institute for Information Technology, OFFIS [10,11]. In contrast to Ptolemy- or HLA-based systems, it features a more concise set of functionalities, and is aimed at being easy to apply for users from different domains.

Due to the complex character of co-simulation frameworks, it can be difficult for CPES researchers to assess which tool suits their needs. This paper is aimed at facilitating this task via a thorough comparison of MOSAIK, as a light-weight tool on the one hand, and an implementation of HLA, as a representation of the popular standard on the other hand. This comparison is based on both, the particular architectural concepts of the tools as well as results from a simulation study implementing a representative CPES setup.

The remainder of this paper is structured as follows: Sect. 2 features a discussion of co-simulation in CPES in general as well as MOSAIK and HLA in particular. The conceptual tool comparison is conducted in Sect. 3, followed by the co-simulation study described in Sect. 4. Section 5, finally, concludes the paper.

2 Co-simulation in cyber physical energy systems

Co-simulation in CPES is a complex topic that involves various aspects. A comprehensive overview of the fundamental concepts is provided in [12]. Some of the most basic concepts are summarized in the following.

A co-simulation setup typically consists of independently executable *simulators* that represent different components/domains of the simulated system. Furthermore, interfaces are implemented to connect the simulators to the framework, as well as a middleware that organizes the communication between the simulators. This middleware can either include a so-called *master algorithm* that organizes the complete co-simulation process—or only a set of communication services so that the co-simulation process emerges from the interaction of the simulators. The temporal synchronization of simulators may either be realized in an iterative manner or non-iteratively (explicitly). Since CPES co-simulation often involves closed source and other black box simulators, explicit coupling schemes are usually more widely applied. In these schemes, simulators are either handled one after the other (*serial*) or in *parallel*.

Since CPES co-simulation involves various different research domains, the demand for standardization is high, especially for the interfacing of simulators. A popular standard is the *Functional Mock-up Interface* (FMI) [13] that allows embedding of simulators in so-called Functional Mock-up Units (FMU). Due to the standard's popularity, it is supported by several tools as well as specifically designed toolboxes (e. g. [14]).

MOSAIK and HLA, the tools discussed in this paper, both follow the presented understanding of co-simulation. Their specific designs are reviewed in the following.

2.1 MOSAIK

The MOSAIK framework has been designed specifically for CPES/smart grid research with a special focus on the flexible creation of large-scale system configurations that may serve for testing of control strategies [15].

The architecture of MOSAIK consists of a simulator management module (*sim-manager*) and a *scheduler*. The sim-manager enables data exchange with simulators by establishing TCP connections with them. The scheduler, on the other hand, coordinates the exchange of data between all connected simulators based on a common simulation clock. In its current state, the scheduler provides discretely-timed, explicit simulator coupling.

The MOSAIK system is completed by two APIs for user interaction: the *Component-API* and the *Scenario-API*. The Component-API has to be implemented for each simulator that is connected to MOSAIK. It sets up a TCP socket and organizes the data exchange with MOSAIK in the JSON format. Various versions of the API are available for different programming languages like Java, Python, or MATLAB. This way, model developers may implement the interface in the language that is most suitable for their simulator. The implementation itself is conducted by providing a *meta description* for the simulator: a high-level description of the provided simulation models as well as their variables. Furthermore, a small number of interface functions have to be implemented that are used by MOSAIK to control the simulator. Note that non-simulator components like database or data analysis systems may be integrated into MOSAIK using the same API. A mapping between the Component-API and FMI also allows the integration of FMUs into MOSAIK co-simulation [16], [17].

The Scenario-API, finally, provides a set of commands that allows users to instantiate model entities from the integrated simulators and establish connections between them. The MOSAIK user is to employ these functions in a *scenario* script that may then be executed to run the co-simulation process.

2.2 High level architecture

HLA is a general-purpose architecture for distributed simulation. It has been developed in the mid-nineties following

an initiative of the United States Department of Defense for the purpose of combat simulation coupling [18]. The first standardization of HLA came in the year 2000 when IEEE declared it as the IEEE standard 1516 for modeling and simulation [19]. Since then, the HLA capabilities have been expanded. Its current version can be found under the IEEE standard 1516.2-2010 [20].

HLA has been designed for coupling of highly autonomous simulators while giving ample control to the user. Using the HLA terminology, these simulators are referred to as *federates* while the entire co-simulation setup is referred to as a *federation*. The communication between federates is realized via a message-bus, the so-called *Runtime Infrastructure* (RTI). Through seven groups of services, HLA provides synchronization of federates and message passing via the RTI. For description of the exchanged data, HLA demands the specification of *Simulation Object Models* (SOM) on the federate level and a *Federation Object Model* (FOM) on the federation level. These structures are defined using the *Object Model Template* (OMT). Finally, HLA specifies a set of ten design rules for the creation of federations (rules 1–5) and federates (rules 6–10). The synchronization of the entire federation depends on the combination of the particular synchronization mechanisms provided by its federates. Thus, various synchronization mechanisms can be implemented with the proper invocation of HLA services, some of which will be tested later in this paper.

HLA is capable of synchronizing time-stepped and discrete event simulators. The first type of simulators are not event-responsive. When a time-stepped simulator reaches a synchronization point, it updates its coupling variables based on the latest message that is intended for it (while any message in between synchronization points except the latest one is not received). The second type of simulator, the event-responsive simulators, is interrupted in its time-progression if there is a message intended for it. Using the corresponding synchronization mechanism, an event-responsive simulator receives all messages that come its way. With HLA, the simulators can choose one or the other type of operation with every step forward that they take.

Finally, the publisher-subscriber mechanism of HLA results in loose coupling between simulators, which allows simple topological reconfiguration of the simulated system during runtime.

3 Conceptual tool comparison

In terms of their conceptual architectures, HLA and MOSAIK display some structural analogies. In both setups, the simulated system is divided into subsystems that are represented by federates (HLA) or simulators (MOSAIK), respectively. The data exchange between the subsystems is managed by a central instance: the RTI in HLA on the one hand, and MOSAIKs software core on the other hand, consisting of the sim-manager and the scheduler. Furthermore, the communication between the RTI and the federates is handled via a number of standardized function calls, similar to the connection between MOSAIK and its simulators.

Aside from these architectural similarities, however, HLA and MOSAIK imply workflows of different nature for their users. There are typically two types of tasks a user may assume when working with a co-simulation environment. The first one is the integration of a new component into the environment. The second one is the creation of a co-simulation study employing the already integrated components. Both of these tasks require different specification and implementation work in HLA and MOSAIK.

Integration of new simulators into MOSAIK requires, as mentioned before, implementation of the Component-API. This involves the specification of the meta description and the implementation of interface functions. For integration of new federates into HLA, on the other hand, a SOM has to be specified and a so-called *federate ambassador* implemented that employs RTI services.

MOSAIK's meta description is a very high-level simulator representation that simply specifies the simulation models that may be instantiated from the simulator, their parameters that may be adjusted by the user, and their attributes that may serve for data exchange with other simulators. No specification of units, data types or variability is given for the attributes (in contrast to the FMI standard). Instead, users are expected to care for attribute compliance themselves when coupling simulators. The SOM of HLA, in contrast, is much more extensive. Similar to the meta description, it includes information about objects and attributes provided by the federate. However, the SOM descriptions are much more detailed, incorporating information about data types, units, resolution, accuracy, and so forth.

For the interaction between MOSAIK and a simulator, four interface functions are needed that have to be implemented by the user. They are then called by the MOSAIK software core to assume the basic tasks of (1) initializing a simulator, (2) instantiating and parameterizing simulation models, (3) providing input and advancing the simulator in time, (4) and requesting the simulator's output data. In contrast, HLA provides a much wider range of functions for the coordination between federates and the RTI which allows for more nuanced interfacing. The functions are not implemented by the user but provided by the RTI as services. The more than 20 services are grouped into clusters like federation management, object management or time management. In order to employ the desired services, the user has to implement the already mentioned federation ambassador that sends calls to the RTI and receives callbacks from it.

Another major purpose of the discussed systems is the create of executable co-simulation setups. For the creation of a co-simulation scenario in MOSAIK, the user has to write a scenario script in Python, employing commands from the Scenario-API. This script should specify which integrated simulators are used, how many model entities are instantiated, and how they are interconnected. Furthermore, data values have to be given for the parameterization of simulators and model entities, as well as the length of the simulated time period. The interconnections may be guided by rules that allow filtering by entity types or previously made connections. This way, even complex connections between large model entity sets can be established with only a few lines of code.

Co-simulation experiments in HLA are less centrally defined than in MOSAIK. As indicated before, HLA federates are more autonomous than MOSAIK simulators, and thus, retain more control over the simulation process. For example, they possess functionalities to dynamically connect to or disconnect from a running simulation. This way, a co-simulation experiment is strongly defined by the configuration of the individual federates. Nevertheless, the HLA specifies a federation also via some central structures. The most important structure for setting up a federation is a FOM. Like the SOM, it is based on the OMT, but while the SOM describes the attributes and objects of a single federate, the FOM describes all attributes and objects that may be used for data exchange within the federation. SOMs are in some sense subordinate to the FOM. In general, a FOM is designed for a specific application domain and can be reused for new simulations in the same domain. Aside from a FOM, federations may be complemented by so-called *federation agreements* or scenarios definitions for documentation purposes, but these concepts are not specified by HLA rules or official templates.

All in all, MOSAIK and HLA both follow the common structure of co-simulation systems possessing individual simulator components that communicate via a message bus. This is reflected in the schematic architectural overview in Fig. 1. Similarities are given in the form of a communication hub (red, striped), component interfacing and description (green, solid), and some specification of component interaction (blue, dotted). Nevertheless, the concepts also differ in some key design principles. As indicated above, a HLA federate is potentially more versatile than a MOSAIK simulator and can intervene with the simulation flow more autonomously. Accordingly, the course of the simulation execution is more loosely defined in HLA than in MOSAIK in order not to limit the capabilities of the federates. While MOSAIK demands a scenario script that specifies fixed interconnections between simulators, HLA merely defines common data structures via the FOM. All other aspects of the federation execution are defined by the federate behaviors and publication-subscription scheme for data exchange.

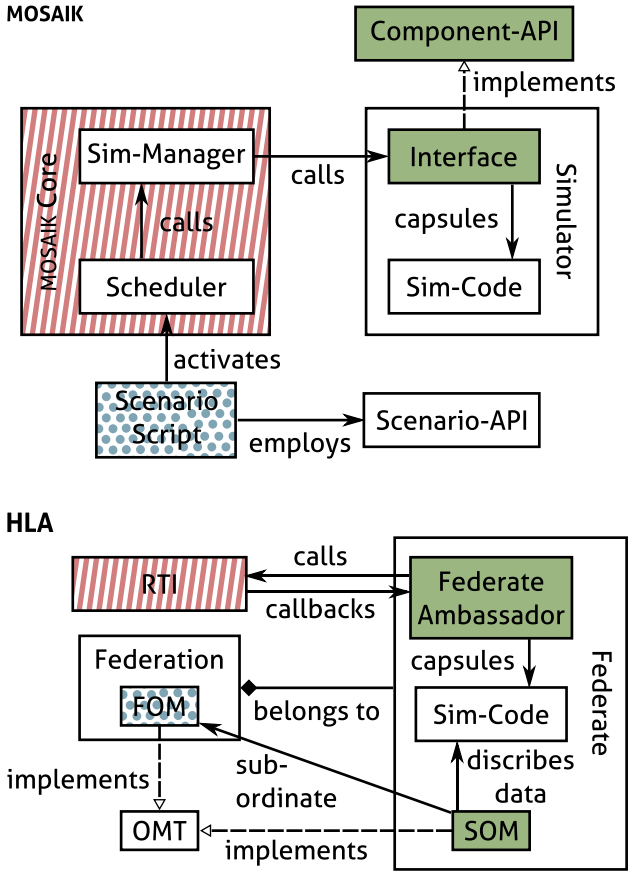

Fig. 1 Architecture overviews of MOSAIK and HLA

The main differences and similarities between MOSAIK and HLA are also reflected in the ten design rules for federates and federations that are part of the HLA standard. The first five rules represent requirements for the design of federations. Their notions are largely mirrored by MOSAIK. The rules 2–4 especially illustrate the architecture of the HLA. They postulate:

- All objects in the federation shall belong to federates and not to the RTI.
- Data exchange during federation execution shall happen exclusively between federates and the RTI.
- Communication between federates and the RTI shall only be executed via the services defined in the standard.

All of these principles are also true for MOSAIK when replacing the terms "federate" and "RTI" with "simulator" and "sim-manager", respectively. This underlines the basic architectural similarity between the two concepts. In contrast, the HLA rules 6–10 specify the federate design and illustrate some of the major differences between MOSAIK and HLA. These rules grant a lot of autonomy to federates. For example, they shall be able to update their conditions for data exchange

Table 1 Overview of comparison between MOSAIK and HLA

Category	MOSAIK	HLA
Communication	Handled by Sim-Manager; simulators are called	Handled by RTI; federates send requests and receive callbacks
Components	Simulators with limited capabilities; user implements interface functions	Very flexible federates; user employs RTI services
Time synchronization	Organized by scheduler; discrete time, variable step size	Individually defined for the federates
Data exchange	Users take care of exchange validity	Validity defined by SOMs and FOM
Simulation configuration	Defined in executable script via Scenario-API	Defined via individual federate message subscriptions

or transfer ownership of objects between each other. Such concepts are not stipulated in MOSAIK. Furthermore, federates have more power over their time management in HLA. Instead of being externally stepped like in MOSAIK, they may request time advancement at the RTI. Depending on the type of request, timing may be negotiated between federate and RTI, allowing for more elaborate options for temporal coupling.

In summary, the conceptual comparison reveals that HLA provides a more versatile co-simulation framework than MOSAIK. Its federates may possess a wider variety of capabilities than MOSAIK simulators so that more different forms of interaction are possible. This is especially true for time synchronization. Since HLA allows for negotiation of time advancement, iterative coupling is possible, which so far is not provided by MOSAIK. However, it has to be noted that the versatility of HLA comes at the cost of a more complex implementation process. Since MOSAIK's Component-API is more concise than its HLA equivalent, its usage is easier to learn. Furthermore, creating an executable co-simulation study in MOSAIK requires only one script with the help of the Scenario-API. The corresponding process in HLA is more loosely defined and typically requires individual configuration of the federates. Therefore, deciding between the usage of MOSAIK and an implementation of HLA entails a trade-off between a flat learning curve (MOSAIK) and a wider field of application (HLA). In addition, it has to be considered whether simulators are to be treated as black boxes or should be freely configurable. The former point follows the MOSAIK philosophy while the latter stands of the HLA design. The conceptual comparison of the two systems is summarized in Table 1.

4 Co-simulation study

It has been stated above that the HLA design allows for more types of simulator coupling than MOSAIK. Therefore, comparing the two systems in the context of a co-simulation study can be misleading when being based on incomparable synchronization schemes. However, if analogous schemes are selected, results produced with MOSAIK and HLA should be equivalent to each other. Studying this hypothesis is the purpose of this section. Two simulators have been coupled using only those schemes that are applicable in HLA as well as in MOSAIK. Such a limitation is reasonable since selection of the coupling scheme is influenced by capabilities and limits of the simulators in question. In other words, simulators are assumed here that are not compatible with HLA's more elaborate coupling approaches and thus may just as well be handled by MOSAIK.

As a HLA implementation the software tool CERTI [21] has been used, version 3.5.1. Just as MOSAIK, it follows an open source licensing model. The employed MOSAIK version is 2.3.0.

4.1 Test system configuration

The capabilities of MOSAIK and HLA will be compared by simulating the storm control of a wind turbine generator. This protection mechanism is engaged when the blade tip speed, and hence rotor frequency violates a threshold value, which commonly equals the rotor speed at rated power output. The test system is shown in Fig. 2. Although such a small system does not call for advanced modeling or simulation methods, the system contains both continuous and discrete behavior, and thus is considered a good and comprehensible example for the comparison of MOSAIK and HLA.

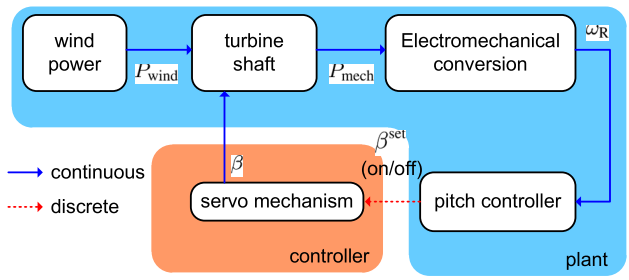

Fig. 2 Elementary wind turbine model (plant) with active pitch regulation (controller)

It consists of a wind power model simulating a sudden wind gust, a turbine shaft model, a second-order plant model of the electromechanical conversion, a discrete pitch controller, and a servo mechanism for blade pitching. In case the rotor speeds exceeds 110% of the rated value, the pitch controller enables the servo mechanism β^{set}. The servo mechanism on its turn gradually increases the pitch angle β, decreasing the turbine shaft power.

Engaging the servo mechanism is a sudden event based on the physical behavior of the wind turbine. From a co-simulation viewpoint, it is interesting to see how such an event is processed between different simulators/federates. Therefore, the test system is split into a plant simulator producing the event and a controller simulator, which responds to it, using β^{set} as an interface attribute.

4.2 Co-simulation setup

The plant as well as the controller simulation model have been created in Modelica and exported as FMUs. A monolithic reference simulation has been conducted using the combination of both models within Modelica.

As mentioned in Sect. 2, the basic, non-iterative coupling between two simulators may either be serial or parallel. These are the only options for simulator coupling supported by MOSAIK whereas HLA may also allow extensions like stepping negotiation or *look-ahead* calculations (if they are supported by the simulators in question). A look-ahead can be implemented in HLA by making the RTI time lag behind the simulator time by one step. More explanation on this topic will follow further below.

Aside from the general synchronization mechanism, the temporal resolution (i.e. the time step size) of the individual simulators plays a crucial role in their coupling. Both, MOSAIK and HLA, allow simulators to employ adaptive time step-sizes. For the sake of comparability, however this example study utilizes a fixed time step-size. Two basic cases of discretely timed co-simulation are examined: (1) both simulators have the same step size, and (2) the simulators have different step sizes. Combining these setups with the two synchronization options, six test cases (TC) have been established and realized with both, MOSAIK as well as HLA (see Table 2). In the serial setup (TC 4-6), the plant simulator is always stepped before the controller.

Table 2 Test cases for system comparison

	Parallel setup	Serial setup
Both step sizes: 0.02 s	TC 1	TC 4
Both step sizes: 0.015 s	TC 2	TC 5
Plant step size: 0.02 s, Controller step size: 0.015 s	TC 3	TC 6

4.3 Results

The behavior of the simulated system is characterized by the wind power (and hence shaft speed) exceeding the specified threshold which leads to a response of the control mechanism. The shaft power (in p.u. of the actual wind power) reaching the turbine is plotted over time in Fig. 3a. It can be seen that the turbine input increases quickly during the first two simulated seconds until the threshold is reached (gray, dashed line). After that, the interference of the controller leads to a decrease of the wind power reaching the turbine.

Next to the threshold crossing, two more events can be defined, related to this system behavior. For one, the information of the threshold crossing reaches the controller to trigger it. The final event is the response of the controller. In the monolithic simulation with a temporal resolution of 0.001 s, these three events are processed within the same time step. In the co-simulation setups, in contrast, this is not necessarily possible since the system is split between the plant and the controller. In fact, the conducted co-simulation

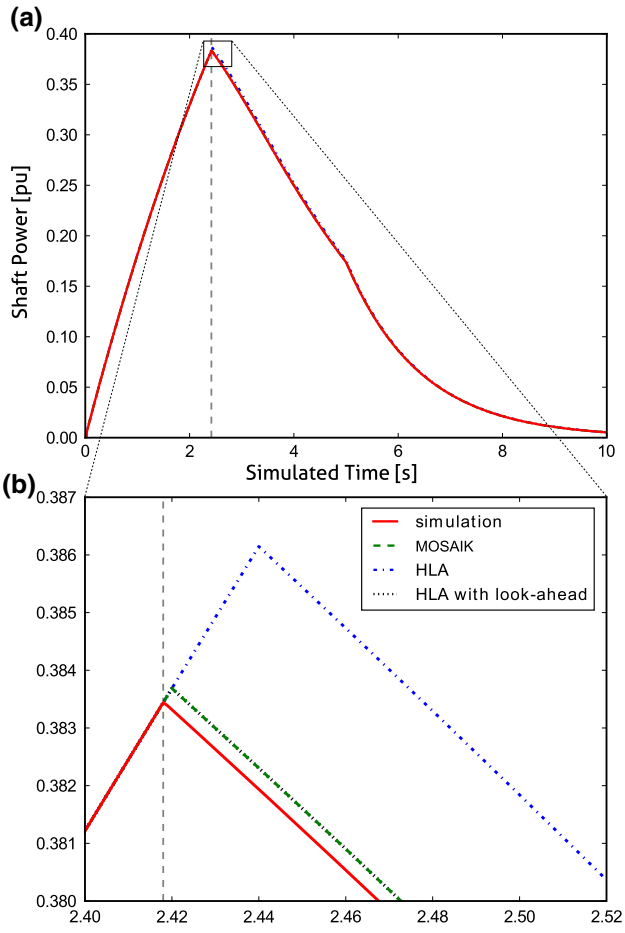

Fig. 3 Calculated wind power input to the turbine for different (co-)simulation setups. Full plot (**a**) and zoomed-in around the time of the event (**b**). Co-simulation is conducted with the setup of TC4

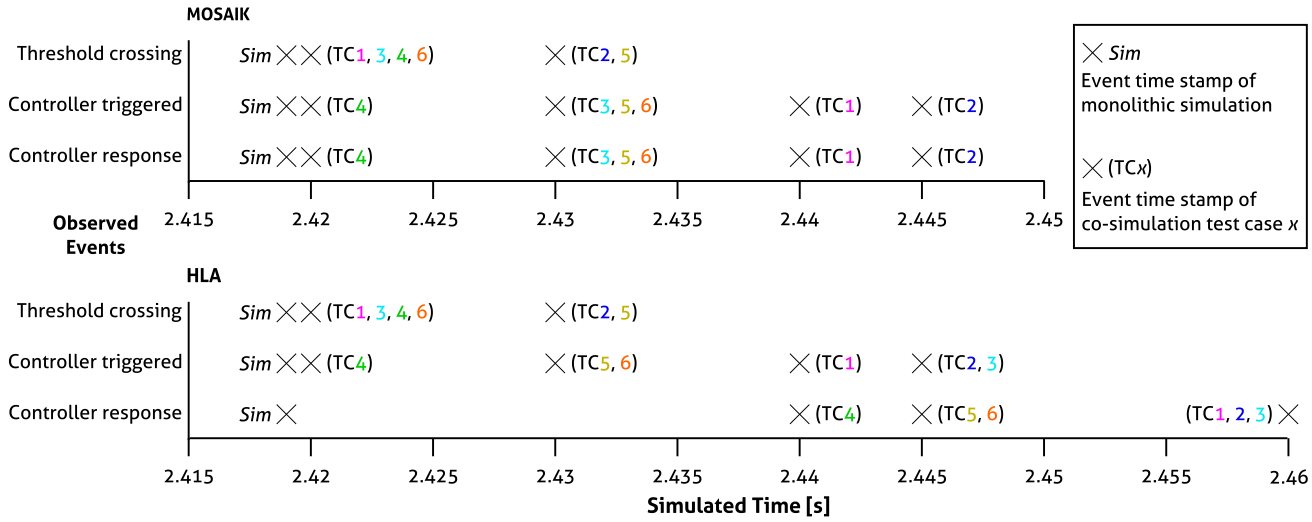

Fig. 4 Timing of events in the conducted co-simulation test cases

experiments reveal that the timing of the events depends on the employed synchronization scheme in combination with the temporal resolution of the simulators. Figure 4 displays the time stamp of the three discussed events for every conducted test case. Each time stamp is indicated by an ×. It can be seen that the time stamps produced by the monolithic simulation (Sim) are 2.419 for all three events. Since this is no multiple of the resolutions of the separate simulators, none of the co-simulation test cases could reproduce these time steps. Even more so, the events cannot be processed within the same time step for some test cases due to the data exchange between the simulators.

A number of interesting observations can be made when comparing the time stamps of the events for the different test cases. First of all, the serial synchronization setup with a time step of 0.02 s (TC4) produces the best output of all the analyzed co-simulation setups. This can be explained by the order of the events. Since the plant simulator registers the first event and then triggers the controller, all events can be processed within the same time step if the co-simulation setup features a serial scheme that advances the plant before the controller. Interestingly, test cases with a time step size of 0.015 s tend to reproduce the timing of the events less accurately despite the higher resolution. This is explained by the fact that the actual event times are closer to a multiple of 0.02 s than of 0.015 s. In other words, due to the system's dynamics, a higher temporal resolution does not necessarily lead to more accurate results—if it is not at least a magnitude higher. Another interesting observation is the fact that all test cases conducted with MOSAIK reproduce the event times slightly better than those conducted with HLA. The explanation for this lies in the type of HLA services used for this comparison. So-called *non-zero look-ahead* services have been used. These services stipulate that a non-zero look-ahead time must be provided by the federate to the RTI in order to step forward in time. Under such rules, the message sent by one simulator to another will take non-zero time to be delivered to its destination. Thus, while mosaik allows for a message to be sent and delivered at the same time instance, the particular set of HLA services used for this comparison adds a delay of one time step between the time the message is sent and the time it is delivered.

The discrepancy between the MOSAIK and the HLA results can also be seen in Fig. 3b (zoom). For both co-simulation systems, the results of TC4 are compared with those of the monolithic simulation (red, continuous line). While none of the co-simulation systems is able to reproduce the simulation results perfectly, the MOSAIK results (green, dashed line) approximate them more closely than the HLA results (blue, dash-dotted line). However, this is only true for a HLA setup that displays similar limitations as MOSAIK. As mentioned before, HLA federates may be equipped with more capabilities than MOSAIK simulators. Introducing the ability to conduct look-ahead operations into the federate design leads to a HLA setup that matches the output of the MOSAIK co-simulation (black, dotted line, that is overlaid on top of the green, dashed line). After all, the look-ahead is able compensate the non-zero look-ahead aspect of the HLA services. Furthermore, it has to be noted that newer versions of HLA include *zero look-ahead* services and thus should be able to match the MOSAIK results without the need to implement look-ahead operations.

5 Conclusion

The presented work gives a concise overview of structural similarities and differences between the HLA and the MOSAIK