Distributed Hybrid Simulation Using the HLA and the Functional Mock-up Interface

Muhammad Usman Awais*, Peter Palensky†, Wolfgang Mueller‡, Edmund Widl§, Atiyah Elsheikh¶
Austrian Institute of Technology, Vienna, Austria.
Emails: (*Muhammad.Awais.fl, †Peter.Palensky, ‡Wolfgang.Mueller.fl, §Edmund.Widl, ¶Atiyah.Elsheikh)@ait.ac.at

***Abstract*—High Level Architecture (HLA) and Functional Mock-up Interface (FMI) are two simulation interoperability standards. The HLA is older, well established and popular in industry. The FMI is a new standard, with plenty of support from the open source community and scientists. In this paper, it is presented how the strengths of both, the HLA and the FMI, can be utilized to realize a distributed hybrid (or heterogeneous) simulation platform. Two different algorithms are proposed for such a platform. To demonstrate the correctness of algorithms, and their performance comparison, a simulation example is chosen from the domain of complex energy systems.**

***Keywords*—High Level Architecture (HLA); Functional Mock-up Interface (FMI); co-simulation; hybrid simulation; heterogeneous simulation; distributed simulation; parallel simulation; continuous simulation; simulation interoperability; DEVS; OpenModelica; Modelica.**

I. Introduction

With growing demand for simulation engineering, numerous Simulation Packages (SPs) have been developed. Most of them are specialized in some specific field or some specific methodology. However, when simulating multi-disciplinary problems, there arises a need to utilize different specialized SPs for their specific problem domain, while allowing the different packages to share information through some mean. The HLA is a standard for interoperability among different SPs. It was presented in year 2000 [1], since then it is being used in industry largely for Distributed Discrete Event Simulations (DDES).

The FMI [2] is closely related to the work on the Modelica specification [3]. It is widely being used as a standard for model exchange among different SPs. The development of the FMI is currently managed by the Modelica Association Project (MAP) [3]. A brief overview and comparison of both the HLA and the FMI can be seen in [4], which also presents the idea of using the HLA and the FMI in Fixed Time Stepped Simulations (FTSS).

In the aforementioned paper the term FMU-Federate was introduced, originating from the fact that a Functional Mock-up Unit (FMU) is a component that conforms to the FMI specification, while in the HLA terminology an individual simulation component is called as a "Federate". So a simulation program or a federate that has the capability of importing an FMU and simulating it over the Run Time Infrastructures (RTI), can be called as an FMU-Federate. If this program can simulate any arbitrary FMU then it may be called as a "generic" FMU-Federate.

The current work presents two kinds of FMU-Federates, based on the simulation algorithms used in them.

1) Fixed Time Stepped: As the name suggests, such an FMU-Federate progresses in fixed time steps. Typically the HLA "Time Advance Request" and "Time Advance Request Available" services are used by it.
2) Discrete Event Based: An FMU-Federate that progresses based on the internal and external events is considered discrete event based FMU-Federate. Typically the HLA "Next Event Request" and "Next Event Request Available" services are used in such an FMU-Federate. To be able to advance time, based on events, it has to look into the future and predict when the next event is going to occur.

The presented platform uses both kinds of FMU-Federates in the "Federation" (the term for whole simulation in the HLA terminology), while the FMUs used in the test case have been generated from continuous models, using OpenModelica [5]. It is a hybrid (heterogeneous) simulation platform, allowing different types of simulations to run in conjunction, e.g. Fixed Time Stepped Simulations (FTSS), Continuous Simulations (CS), and Discrete Event based Simulations (DES). Moreover, the FMU-Federates used in the test case are "generic", unless there was a tool limitation. Additionally, due to the fact that the platform uses the RTI as a regulator or master, it can be used in any distributed computing environment, including clusters, grids and clouds.

The rest of the document is organized as follows. Section II discusses related work, then the two algorithms that are used in the proposed solution, are discussed in section III. Section IV discusses the test case in detail, and presents comparative results. The last section V entails the conclusive remarks.

It is important to mention that the concepts described in the section III largely depend on the understanding of, specially the HLA, and preferably the FMI. Even if one does not have an in depth knowledge of the FMI, one must have an understanding of continuous simulation. Secondly, the reader should note that in the test case presented here FMUs are continuous simulation components. They were implemented and generated by OpenModelica [5], yet any FMU generated by any tool may be used. Other sections do not require such an in depth knowledge.

II. Related Work

According to the knowledge of authors, there has been no attempt to utilize the HLA and the FMI in conjunction,

978-1-4799-0223-1/13/$31.00 ©2013 IEEE

except of [4]. However, there has been considerable amount of work done to realize heterogeneous or hybrid simulation environments. The Discrete EVent Specification (DEVS) community has done significant amount of work on this topic. In [6] d'Abreu and Wainer have discussed different techniques proposed by the DEVS community to convert a continuous state system into a DEVS model. Zeigler et al. have presented their solution for using DEVS components with the HLA [7]. Readers knowledgeable of both the DEVS and the FMI, will realize that the approach presented here is also applicable to DEVS components, and it does not suffer from the flaws of the technique presented in [7]. Comparison to previous solutions comes in the section III-B2

Similarly Ptolemy II [8] is another project providing a heterogeneous simulation platform. In [9] Ptolemy II and DEVS are compared. However, intuitive representation and GUI makes the Ptolemy II a better choice for users not accustomed to the DEVS formalism.

III. Algorithms and Models

Different aspects of two proposed algorithms will be discussed, in the current section. First the simpler algorithm will be discussed, i.e. "Fixed Time Stepped Algorithm" and then the "Discrete Event Algorithm" will be presented. Both of these algorithms come from the concept called as "zero lookahead based simulations", using the HLA. Fujimoto [10] first presented the concept. This was a very abstract idea for special types of simulations, but the research community has not presented considerable examples of its practical use. Current work partially fills this gap.

A. Fixed Time Stepped Algorithm

Before presenting the algorithms, lets have a look at an ordinary approach towards fixed time stepped simulation using the HLA. The ordinary use of the HLA involves a value called "lookahead". This is the minimum amount of time, before which an active federate promises not to generate any new events. For example if a federate F with lookahead value l_a, requests a time grant for time t_0, then it must not generate any events before $t_0 + l_a$. The naive approach for simulating a continuous model would assume a small time as lookahead, and would keep progressing in fixed time stepped fashion, using "Time Advance Request" service of the HLA. Please see Algorithm 1

In Algorithm 1 lines 3–6 represent a typical fixed time advancing federate behavior. Afterwards the updates should be applied (line 8). Each update sent from other FMU-Federates must be applied in time, so before applying the updates the FMU should be moved forward to the time of update. Afterwards by calling $fmiSetValue$ function the state variables of the model can be set. It is important to note that to ensure correct execution, the updates should be sent in "time stamped order". Lines 9–12 ensure that the model is progressed to the granted time plus the lookahead value. It is important to add the lookahead value, otherwise the update sent to the RTI will not be valid and will generate an exception. At the end, the updated states of the model are sent to the RTI. It is assumed here that the simulation will start from $time = 0$ as this is the case for most FMUs in practice.

Algorithm 1 NaiveTimeStepped (FMUFederate model, RTIAmbassador rtia, Time SimulationEndTime, TimeInterval step, TimeInterval lookahead)

```
1:  time = 0
2:  while time ≤ SimulationEndTime do
3:      rtia.timeAdvanceRequest(time + step)
4:      while (Time is not granted) do
5:          ▷ Process callbacks from the RTI.
6:      end while
7:      ▷ Now time = time + step
8:      ▷ Apply all accumulated updates, if any.
9:      time_1 = time + lookahead
10:     if model.time < time_1 then
11:         ▷ Integrate model to time_1
12:     end if
13:     ▷ Send updated states of model to the RTI.
14: end while
```

1) Drawbacks: The important point to be noted in Algorithm 1 is, the lookahead value. In fact, this is not a problem when all the federates have same execution algorithm, with same step size and lookahead value. The problem becomes visible, when there is "heterogeneous" simulation, or there are more than one type of federates, with different step sizes.

Suppose there are two federates; one time stepped F_{fts} and the other discrete event based F_{des}. Now suppose that F_{fts} has a lookahead value la_{fts}, and F_{des} has lookahead value la_{des}, such that $la_{des} \ll la_{fts}$. Suppose F_{des} is a federate that turns a switch on and off in F_{fts}. Now F_{des} wants to turn the switch on and off at times t_{on} and t_{off}, such that $t_{on} < t_{off}$ and $t_{off} - t_{on} < la_{fts}$. In this case F_{fts} will have to wait for F_{des} until it produces both events. In case there is a feedback loop from F_{fts} to F_{des} and the feedback has to be transmitted during the interval $[t_{on}, t_{off})$, then it is clear that there will be a loss of information, because F_{fts} will never be able to process event at t_{on} before t_{off} occurs, and will never be able to transmit the effect of t_{on} to F_{des} in time.

One solution to this problem is to use very small lookahead values, but it has been proven that a smaller lookahead value is detrimental to parallel execution of simulation [11]. Another solution is to vary the lookahead values during execution, which is an even worse solution, due to two reasons. First, the process of changing the lookahead value is not straightforward. The RTI has to ensure guarantees given to other federates before it really changes the lookahead value. Changing the lookahead value during execution will also cause negative effects on performance for the same reason, that the Lowest Bound Time Stamp (LBTS) will have to be calculated repetitively [11].

2) Improvements: The solution to the above problem lies in so called "episodic simulation" [10]. In such simulations federates are allowed to have a zero lookahead value, and federates are allowed to have different step sizes. The main idea is to use the "Time Advance Request Available" service of the HLA. For the detailed difference of the "Time Advance Request Available" and the "Time Advance Request" services, reader is advised to look into [1]. One distinct difference is that a federate can send and receive multiple events at the granted time, when the time is requested using the "Time Advance Request Available" service.

Algorithm 2 FixedTimeStepped (FMUFederate model, RTIAmbassador rtia, Time SimulationEndTime, TimeInterval step)

```
1: time = 0
2: while time ≤ SimulationEndTime do
3:     ⇛ Start of episode.
4:     rtia.timeAdvanceRequestAvailable(time + step)
5:     while (Time is not granted) do
6:         ▷ Process callbacks from the RTI.
7:     end while
8:     ▷ Now time = time + step
9:     ▷ Apply all accumulated updates, if any.
10:    if model.time < time then
11:        ▷ Integrate model to time
12:    end if
13:    ▷ Send updated states of model to the RTI.
14:    ⇛ End of episode.
15:    rtia.timeAdvanceRequest(time)
16:    while (Time is not granted) do
17:        ▷ Process callbacks from the RTI.
18:    end while
19:    ▷ Apply all accumulated updates, if any.
20: end while
```

Although seemingly Algorithm 2 is similar in structure to Algorithm 1, yet there are semantic differences in Algorithm 2. First, there is no concept of lookahead in Algorithm 2. In common scenarios, at the start of each episode the RTI will look what is the minimum time requested, and it will be granted to the requesting federate. The federate with greater time requested will have to wait until others have passed its requested time. Variable step sizes are possible but it is not advised to use "Time Advance Request Available" service for this. It is recommended to use "Next Event Request Available" service for federates with variable step sizes. The Algorithm 3 is related to this concept.

When federates are demanding equal time advances then all of them will not only keep in sync all the time, but at the end of each episode they will still have the chance to exchange information. The updates produced by a federate running on a slower machine, are always guaranteed to be delivered before the end of each episode.

B. Discrete Event Algorithm

Previously, it was mentioned that the variable step sizes may cause trouble while using Algorithm 2. The solution is to use the "Next Event Request Available" service. For the detailed differences, the reader is advised to consult [1]. An important difference is, when the service "Next Event Request Available" is called with a requested time t_0, then t_0 is just taken as a consideration, and the granted time can be anything less than t_0. After the time grant federate is allowed to request a new time $\bar{t_0}$, only if the federate has received any time stamp ordered updates. The new requested $\bar{t_0}$ may be $\bar{t_0} \gtreqless t_0$.

1) Peculiarities: The main difference of Algorithm 3 is, it can use variable step sizes. After each time grant the federate using Algorithm 3 will predict the time for the next event, this is shown in line 6. The $predictNextEvent$ function can take many forms, but intrinsically it should only analyze the values produced by the model so far and should predict the time when

Algorithm 3 DiscreteEvent (FMUFederate model, RTIAmbassador rtia, Time SimulationEndTime)

```
1: time = 0
2: while time ≤ SimulationEndTime do
3:     getNewPredictedTime = True
4:     while True do
5:         if getNewPredictedTime = True then
6:             eventTime ← predictNextEvent(model)
7:         end if
8:         ⇛ Start of episode.
9:         rtia.nextEventRequestAvailable(eventTime)
10:        while (Time is not granted) do
11:            ▷ Process callbacks from the RTI.
12:        end while
13:        ▷ After time grant time ≤ eventTime
14:        ▷ Apply all accumulated updates, if any.
15:        if model.time < time then
16:            ▷ Integrate model to time
17:        end if
18:        if time = eventTime OR updates ≠ φ then
19:            ▷ Send updated states of model to the RTI.
20:            getNewPredictedTime ← True
21:            break ▷ out of infinite loop to end the episode
22:        else
23:            getNewPredictedTime ← False
24:        end if
25:    end while
26:    ⇛ End of episode.
27:    rtia.nextEventRequest(time)
28:    while (Time is not granted) do
29:        ▷ Process callbacks from the RTI.
30:    end while
31:    ▷ Apply all accumulated updates, if any.
32: end while
```

the next ""significant change will occur in the model. By and large this is an extrapolation function, but the crucial point is to decide what is "significant" while determining the change. In this configuration, a fixed time stepped federate would simply add its time step into the granted time to get the time for next event.

If we were using Algorithm 2 then it would mean that we are assuming the significant values of all the FMU-Federates to occur after a fixed interval. This is similar to having "samples" of a continuous signal. The sampling technique will suffer from over and under sampling if the considerations of Nyquist are not followed. The problem here is different though, in simulations one may or may not know the Fourier transform of generated signal beforehand, and may not be able to select the optimal sampling time.

The solution as proposed in [6] is to use "quantization" instead of sampling. Quantization can take many different forms. A simple quantization technique would be to round the values to some significant precision. More complex forms can be seen in any text book for signal processing e.g. [12]. Though, there is one important aspect in this regard, in the signal processing community whenever there is a technique mentioned for quantization, there is also a focus on removing noise. This is not the case here, as there is no noise in simulations, unless generated on purpose.

Second important line in Algorithm 3 is line 18. Here two things are checked before allowing the federate to ask for the next event. One, if the granted time is equal to the requested time ($time = eventTime$), then it is obvious to request a new time grant. If this is not the case, then the federate should only request a new time (different from before), if and only if, federate has received some state update from the RTI before the recent time grant ($updates \neq \phi$). If both of these cases are not valid, then the federate should request the time it requested before. Effectively, which also means that the federate does not need to get out of its episode.

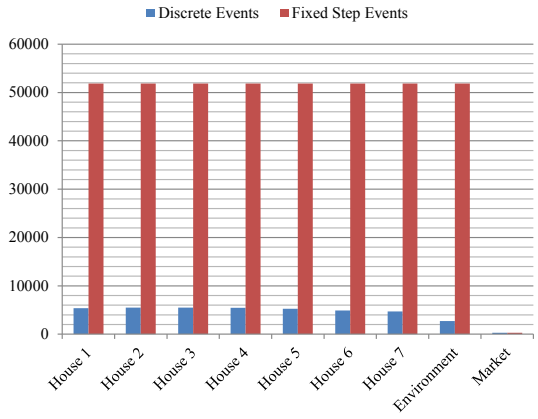

Fig. 1: Number of events generated by Algorithm 2 and Algorithm 3

Significant advantage of Algorithm 3 is the reduced number of events generated by it. In normal scenarios the number of events generated by Algorithm 3 will be less than Algorithm 2. For example Fig. 1 compares the number of events generated in the sample test case. More details of the test case follow shortly, here it is important to note that the number of events generated by Algorithm 3 are a fraction of *10* as compared to Algorithm 2. Reducing the number of events by such a significant amount is specifically advantageous for purely distributed simulation, when the federates reside on distant machines. Reducing the number of events, means reducing the communication overheads and achieving significant performance gains.

2) Comparison to Previous Solutions: The concept of episodic simulation is not new, the need for it has been felt before. Here two systems are presented that tried to achieve the same goal of episodic simulations but they achieved it in the wrong way. The first example is of Ziegler et al. [7] where they are not using "available mode" services of HLA. It is argued that instead of introducing small fixed time steps for internal synchronization, multiple episodes could have been used, which would remove discrepancies in the results.

EPOCHS [13] is another example where hybrid systems have been made to run in conjunction. The researchers used synchronization points to achieve this, which are very costly in terms of execution. Their ineffective use produced many errors in EPOCHS. Here the episodic execution could have been used, which not only had improved the accuracy but also it would have been much faster in execution.

IV. Test Case and Results

To demonstrate the correctness of the above presented algorithms, a test example has been chosen. The choice was made based on the fact that the example has been tested with many different tools. See [14] and [15] for the details of problem and the tools used for simulating the problem. Here only a brief overview of the problem will be presented.

A. Model Description

There are three main components of the model

1) House : Consists of volume, a heater with controller and an agent.
2) Environment temperature: A simple fluctuation of ambient temperature.
3) Market: Component regulating the price of electricity based on the consumption.

A "house" is a simple thermal system, which consumes energy in order to maintain its temperature. An "agent" is responsible for setting the maximum and minimum temperature threshold of the house, based on the price of energy. There is a heater, which is responsible for heating the house, it is controlled by a controller, which gets its minimum and maximum temperature threshold from the agent. The volume of the house has a significant effect on its temperature profile. Each house is also attached to the ambient temperature, coming from "environment". The "environment" is a very simplified model, based on simple sinusoidal fluctuation of temperature. The "market" defines the price of energy based on the total consumption of the houses. Fig. 2 shows the overview of the model. For details of the model and a mathematical description see [15].

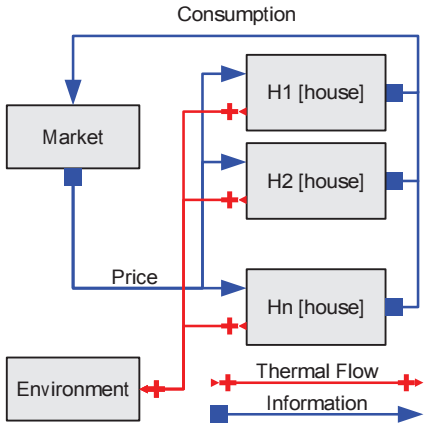

Fig. 2: Model overview (taken from [14])

In order to use the full functionality of the HLA RTI, there is no option but to use a commercial version of the RTI. No open source RTI currently offers full support for "available mode" services. The academic version of commercial RTIs offer a limited number of federates, hence there are only 7 houses simulated to produce results.

Significant values of the model are enumerated below, which form the basis of comparison.

1) Average inside temperature of all the houses.
2) Price fluctuation, based on the energy consumption.

B. Types of Federates and Simulations

Three runs of the sample simulation were executed. One with OpenModelica, as a reference for other simulations, its results are assumed to be prefect. Second, with Algorithm 3 functioning as the main algorithm for all federates. Third, where all of the federates were running Algorithm 2, except the "market" federate. The "market" federate could not be used as a time stepped federate due to a limitation in OpenModelica. The FMU for "market" was not producing the correct results, presumably due to the fact that it was a piecewise continuous function.

As described earlier, all the components were first implemented and simulated using OpenModelica. Then the same components were exported as FMUs, except the "market" component. The "market" component took a reading of all consumptions after each 15 minutes, took the averages and calculated the new price. In the generated FMU a fluctuation of values, was not occurring. To replace this FMU, a federate was implemented by hand, which acted identical to the prescribed definition of the "market". Because its step size had to be 15 minutes, in comparison to 5 second step size of rest of the federates, so it was implemented as a "discrete event federate" as advised in section III-A2.

The initial values for all the simulations were identical and fixed, so the results are reproducible. The simulation was run for 3 days of simulation time, which is 259200 seconds. One unit of simulation time represented one second. For the integration of FMUs, any step size could be chosen, even a fraction of a unit was also allowed. The communication among FMU-Federates took place after 5 seconds of simulation time, this is only valid for FMU-Federates using Algorithm 2. FMU-Federates using Algorithm 3 could vary their step size, so they could send updates after arbitrary time intervals.

C. Results

Here the results of all simulations are presented comparatively. Before comparing the results it is important to have a look at Fig. 3, it shows the ambient temperature. It is a simplistic sinusoidal fluctuation, which governs all the values in the simulation.

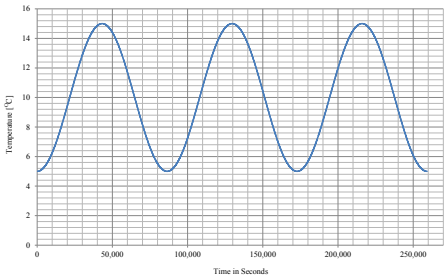

Fig. 3: Environment profile

In both figures, Fig. 5 and Fig. 4, upper most figure is for Algorithm 2, middle one for Algorithm 3 and the bottom one is

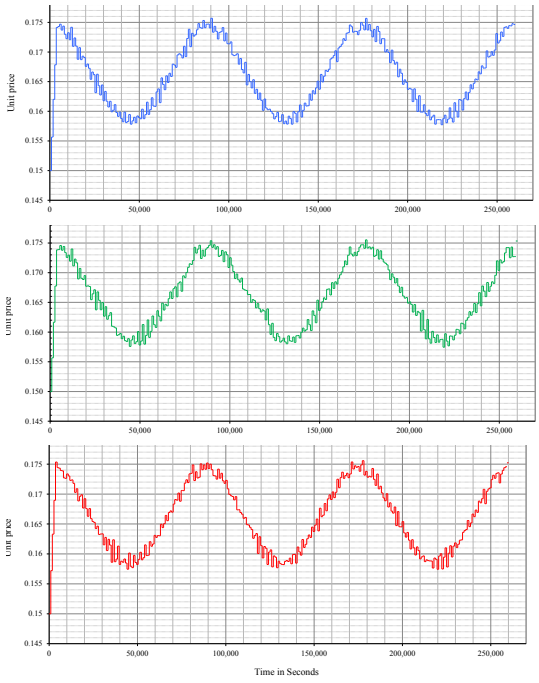

Fig. 4: Results for variable "Price"

for OpenModelica. For variable "Price", the standard deviation of Algorithm 3 from OpenModelica results is 0.0012, while for Algorithm 2 it is 0.00108. For variable "verage Temperature", the standard deviation of Algorithm 3 from OpenModelica results is approx. 1.13, while for Algorithm 2 it is approx. 1.09.

It is quite clear from the results that all simulations are behaving similarly. The minimum and maximum values are also the same. In reality there was no difference in "market" component using Algorithm 2 or Algorithm 3, in both cases the price was being updated after 900 seconds (15 minutes), and the same code was used for both fixed step and discrete event simulations. Only the consumption behavior of the houses could make any difference. The small difference in the standard deviation of both the results proves that there was no significant change in the consumption behavior of the houses, even with changed algorithms.

The calculation of the standard deviation was easy in this case, as all three simulations produced results at precisely the same time intervals. This was not the case for Fig. 5, where approximations had to be used. As OpenModelica, Algorithm 2 and Algorithm 3 calculated values at completely different intervals, so for the sake of comparison nearest intervals were chosen. Hence, the value of standard deviation is not hundred percent correct, it also includes the approximation error.

The average temperature profile in Fig. 5 also shows the same behavior for all the simulations. Although the boundary values of the simulations differ a little, but it is clear that the main statistical information of the model does not change by changing the simulation platform and algorithms used for simulation.

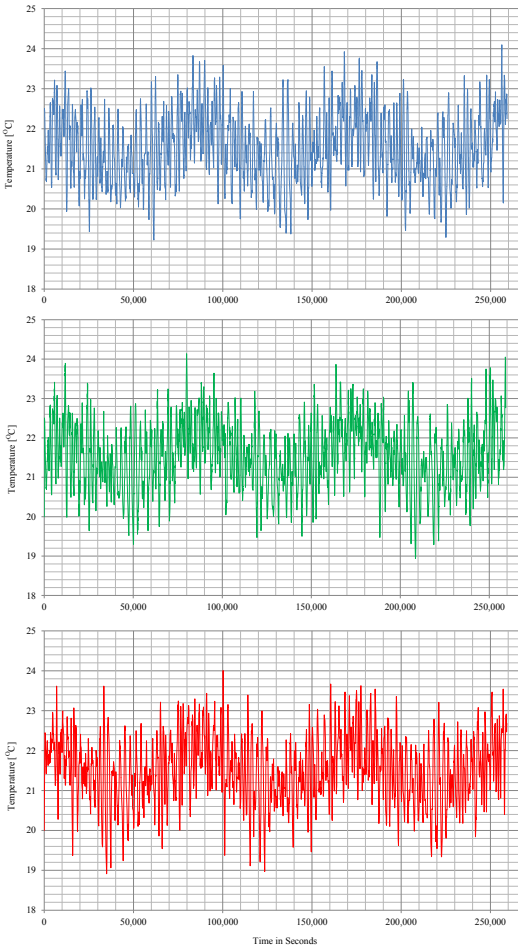

Fig. 5: Results for variable "Average Temperature"

V. Conclusion

The paper presents a novel idea of using the HLA in conjunction with the FMI specific FMUs. Two different algorithms have been presented. It is shown that the algorithms presented are suitable for distributed hybrid simulation, where individual simulation components may be discrete event based, fixed time step based, or continuous simulation components. A test case has been presented that shows the correctness of results, taking results generated by OpenModelica as a reference.

The test case was designed to use FMUs pertaining to continuous simulation domain, as the challenges of simulating continuous models in distributed environments are much greater than DES or FTSS models. One FMU-Federate in both simulations was a discrete event FMU-Federate, which used a fixed interval for the time advance (900 seconds). It was not a "generic" FMU-Federate, as its implementation was done separately. This was the "market" component, which could not be generated as an FMU due to the limitations in the OpenModelica FMU generation process.

For variables which show mixed behavior, having regions where the solution curve displays much variation, and is smooth otherwise. Algorithm 3 can vary its step size in different regions, provided the prediction method used to predict future events is well suited, while Algorithm 2 (fixed step) will have to shift to quite small step sizes, causing performance issues, by raising the number of events. In the presented test case, the number of events generated by Algorithm 3 (discrete event) were a fraction of 10 as compared to Algorithm 2 (fixed step).

It is important to note that in the examined test case, all input to output relationships are stable and smooth. In cases where a small change in an input can cause big changes in the system, achieving stability can be challenging. This is one of the areas of our further research.

References

[1] S. I. S. Committee *et al.*, *IEEE standard for modeling and simulation (M&S) high level architecture (HLA)-IEEE std 1516-2000, 1516.1-2000, 1516.2-2000.*, Institute of Electrical and Electronics Engineers Std., 2000.

[2] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold *et al.*, "The functional mockup interface for tool independent exchange of simulation models," in *Modelica'2011 Conference, March*, 2011, pp. 20–22.

[3] Modelica.org. [Online]. Available: https://www.modelica.org/projects

[4] M. Awais, P. Palensky, A. Elsheikh, E. Widl, and M. Stifter, "The high level architecture RTI as a master to the functional mock-up interface components," in *ICNC 2013 International Workshop on Cyber-Physical System (CPS) and its Computing and Networking Design*. IEEE, 2013.

[5] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. Wiley-IEEE Press, Sep. 2011.

[6] M. C. D'Abreu and G. A. Wainer, "Models for continous and hybrid system simulation," in *Winter Simulation Conference - WSC*, 2003.

[7] B. Zeigler, G. Ball, H. Cho, J. Lee, and H. Sarjoughian, "Implementation of the devs formalism over the hla/rti: Problems and solutions," in *Simulation Interoperation Workshop (SIW)*, no. 99S-SIW, 1999, p. 065.

[8] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity-the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[9] S. Kim, H. Sarjoughian, and V. Elamvazhuthi, "Devs-suite: a simulator supporting visual experimentation design and behavior monitoring," in *Proceedings of the 2009 Spring Simulation Multiconference on ZZZ*. Society for Computer Simulation International, 2009, p. 161.

[10] R. M. Fujimoto, "Zero lookahead and repeatability in the high level architecture," in *Proceedings of the 1997 Spring Simulation Interoperability Workshop*. Citeseer, 1997, pp. 3–7.

[11] K. Perumalla, "Parallel and distributed simulation: traditional techniques and recent advances," in *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 84–95.

[12] A. V. Oppenheim, R. W. Schafer, J. R. Buck *et al.*, *Discrete-time signal processing*. Prentice-hall Englewood Cliffs, 1989, vol. 2.

[13] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *Power Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 548–558, 2006.

[14] E. Widl, P. Palensky, and A. Elsheikh, "Evaluation of two approaches for simulating cyber-physical energy systems," in *Proceedings of the 38th IEEE Conference on Industrial Electronics IECON 2012*, Montreal, Canada, 2012.

[15] A. Elsheikh, E. Widl, and P. Palensky, "Simulating complex energy systems with modelica: A primary evaluation," in *2012 6th IEEE International Conference on Digital Ecosystems Technologies (DEST)*. IEEE, 2012, pp. 1–6.