HLA high performance and real-time simulation studies with CERTI

Jean-Baptiste Chaudron, Martin Adelantado et Eric Noulard
Office National d'Etudes et de Recherches Aérospatiales
email: name.lastname@onera.fr

Pierre Siron
Institut Supérieur de l'Aéronautique et de l'Espace
email: name.lastname@isae.fr

KEYWORDS
Real-time simulation, HLA, RTI, embedded systems.

ABSTRACT

Our work takes place in the context of the HLA standard and its application in real-time systems context. Indeed, current HLA standard is inadequate for taking into consideration the different constraints involved in real-time computer systems. Many works have been invested in order to provide real-time capabilities to Run Time Infrastructures (RTI). This paper describes our approach focusing on achieving hard real-time properties for HLA federations through a complete state of the art on the related domain. Our paper also proposes a global bottom up approach from basic hardware and software basic requirements to experimental tests for validation of distributed real-time simulation with CERTI.

Introduction

Modern systems become more and more complex with an increasing number of both components and interactions between them. These different applications often require their services to be delivered within a given amount of time (deadline). This focus is the problematic of real-time system which are defined as those systems in which the correctness of the system not only depends on the logical results of computation, but also on the time at which these results are produced (1). Real-time systems are broadly classified into two categories based on the nature of the deadline, namely, hard real-time systems, in which the consequences of not executing a task before its deadline may be catastrophic and soft real-time systems, in which the utility of results produced by a task with a soft deadline decreases over time after the deadline expires. Examples of typical hard real-time systems are flight control and nuclear power-plant control. Telephone switching system and image processing applications are examples of soft real-time systems.
Distributed computing paradigm proposes a high performance solution thanks to advances in network technologies. Different programs located on several computers interact all together in order to achieve a global common goal. However, designers and developers of distributed software applications have to face several problems such as heterogeneity of the various hardware components as well as both operating systems and communication protocols. Development of middleware standards like CORBA (2) allows to consistently face these problems. The term middleware describes a software agent operating as an intermediary between distributed processes (*Cf.* Figure 1). This software must be considered in the domain of interoperability; it is a connectivity software which enables the execution of several interacting applications on one or more linked computers.

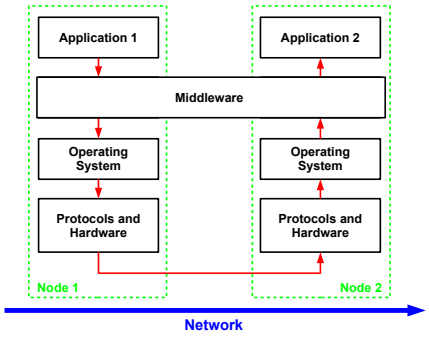

Figure 1: illustration of Middleware

Indeed, real-time experts investigate distributed computing solutions to ensure real-time behavior for complex systems (3). However, traditional distributed standards and middleware architectures could not yet take into account real-time constraints. Real-time aircraft software and hardware embedded components interconnected with middleware have led to some particulars research projects like ARMADA (4) and MIDART[1] (5) and also some advances in current standards to include real-times properties, like Real-time CORBA (6) or more recently DDS (7). The main objective of our work is to use an HLA middleware, compliant with current HLA IEEE 1516-2010 standard (8)(9)(10), to develop, interconnect and maintain real-time simulations of embedded system (hardware-in-the-loop system or

[1] *MIDdleware and network Architecture for Distributed Real-Time systems*

fully simulated system). This article explains how we proceed to ensure real time behavior for our simulations. The use of a distributed simulation architecture to study distributed embedded systems should provide a more natural and flexible framework for new researches in the domain.
The paper is structured as follows: Section 2 describes the problem statement. We present the targeted applications, a background on HLA use for real-time and we describe in detail the CERTI architecture. Section 3 outlines our global approach for real-time simulation purpose. We describe all the techniques and methods used to ensure the correct temporal behavior of the simulator. Different experimental results obtained on our specific platform are illustrated in Section 4. Finally, a discussion of results, as well as currently planned extensions of the infrastructure, is proposed in conclusion.

Problem Statement

Targeted applications

Our work takes place in a global project named PRISE[2]. The main focus of this project is to study new embedded system concepts and techniques through a special hardware and software environment (*Cf.* Part). All these simulations could also be *Hardware-in-the-loop* simulations by connecting real actors: actuators, sensors or real embedded computers in the simulation loop. Obviously, these simulations could also be *Human-in-the-loop* simulations but we are focusing here on real-time aspects.
A collaborative study between Onera and CNES laboratory gave first elements to understand the use of the HLA standard and CERTI run-time infrastructure for real-time simulations (12). The case study, is a satellite formation flying simulation that is made up by four components that are embedded systems simulators for two satellites as depicted by figure 2: ***Federate 1*** is a simulator of the board computer on satellite 1; ***Federate 2*** is a simulator of the board computer on satellite 2; ***Federate 3*** is a simulator of the dynamics of the satellite 1 and finally ***Federate 4*** is a simulator of the dynamics of the satellite 2;

HLA real time background

Simulation is a well established technique used in the man-machine system area for training, evaluation of performance and research. However, works to include real-time specifications and properties to HLA standard are less advanced than others ones (13). We claim that the choice of a distributed computing standard and its underlying middleware is an important starting point to

[2] *Plate-forme de Recherche et d'Ingénierie des Systèmes Embarqués*

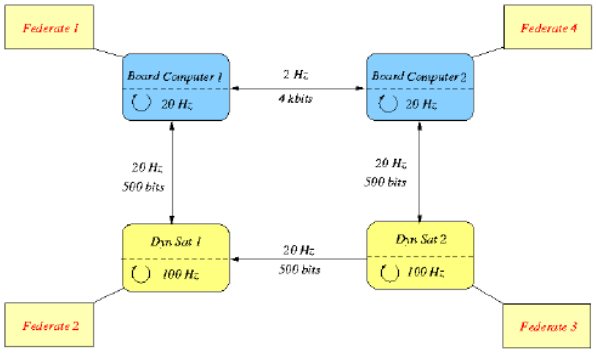

Figure 2: CNES Satellites formation flying simulation

obtain high fidelity, valid and scalable real-time simulations. This choice implies which operating system, which programming language and which hardware could be used for compliance with the middleware. Many studies and integration simulations are elements of the Airbus industrial process (14) but the different models are proprietary (and sometimes certified) as well as the Run Time Infrastructure (RTI, the HLA underlying middleware). The RTI is the distributed software for interconnecting various federates to a global federation execution. The RTI-NG (15) was the first run-time infrastructure developed and used by the US Department of Defense; this RTI is no longer maintained. Since then, several approaches have been investigated to add real-time properties to HLA standard and underlying software RTI:

1. Multi-threaded synchronous process for RTI (16)(17)(18);
2. Global scheduling services in RTI (16)(17)(18);
3. Real-time Optimized RTI services like time Management from Fujimoto and McLean (16) or Data Distribution Management for Boukerche works (18);
4. Quality of service communication with, for example, RSVP[3] (13) or specific protocols like VRTP[4] (19);
5. Use a real-time operating system to allow preemptive priority scheduling (20).

These different techniques allow an improved use of system resources, better scalability and also a higher reactivity of services provided by the RTI. However, no work proposes a complete analysis from simulation requirements to implementation. Most of all, the run-time infrastructure used is never clearly presented (except for (13)(17) by using RTI-NG).

[3] *Ressource ReSerVation Protocol*
[4] *Virtual Reality Transfer Protocol*

Bottom-Up approach (Actions Levels)

The temporal properties of distributed real-time simulation are obtained from a complex combination of the application structure, the used HLA middleware and specific distributed algorithms, the software infrastructure (operating systems and communication protocols) and finally the physical infrastructure (type of computers, type of networks and distribution topology). The specific PRISE platform architecture is composed of:

Hardware: 4 real-time nodes with Opteron 6 core processors, 2 Graphical HP station computer with Intel Xeon processors and high performance GP-GPU, an ethernet Gigabit switch on a dedicated network and also two input organs (Yoke/Throttle/Pedal systems). This global system also proposes a particular advantage, a distributed clock technology allowing same clock reference to each node (21).

Software: Linux Red Hawk (22) Operating system compliant with POSIX Real time standard (23). This RTOS has been already used in the simulation domain by TNO laboratory which uses this OS to run their own RTI implemented in C++. Their experiments have concluded that this operating system is suitable for real-time computing (20).

Middleware: In our approach, we will rely on our Open Source RTI called CERTI because we have a complete control on its implementation.

CERTI

For years, the French Aerospace Laboratory (ONERA) has been developing his own Open-Source middleware RTI compliant with HLA standard called CERTI (24). This RTI runs under several operating systems including Linux and Windows. It is recognizable through its original architecture of communicating processes (*Cf.* Figure 3). Each federate process interacts locally with an RTI Ambassador process (RTIA) through a Unix-domain socket (equivalent to LRC[5]). The RTIA processes exchange messages over the network, in particular with the RTIG process equivalent to CRC[6]), via TCP (and also UDP) sockets, in order to run the distributed algorithms associated with the RTI services.

The CERTI has, originally, no mechanism for taking into account quality of service and no tools to provide an end to end predictability. In this sense, it does not handle events differently according to a priority and it uses no predictability mechanism whatsoever at the network or the operating system. In our case, a key benefit is to master the implementation of RTI used and thus able to incorporate changes in the source code to ensure temporal predictability of CERTI.

[5] *Local Run-time Component*
[6] *Central Run-time Component*

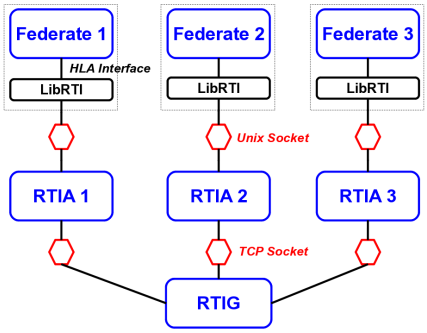

Figure 3: CERTI architecture

Our approach

Towards periodic federates

The concept of periodic federates, named "*repeatability within simulations*" has been introduced by Fujimoto and McLean (25) (26) with their works on real-time and distributed simulations. Federates, involved in this kind of simulation, repeat the same pattern of execution periodically with a time step noted Δ_t. During each step, federates carry four phases: a reception phase, a computation phase, a transmission phase and a slack time phase. ONERA and CNES studies (12) show the necessity of explicitely adding a synchronization phase to ensure the global coherent run time of the whole simulation (*Cf.* Figure 4).

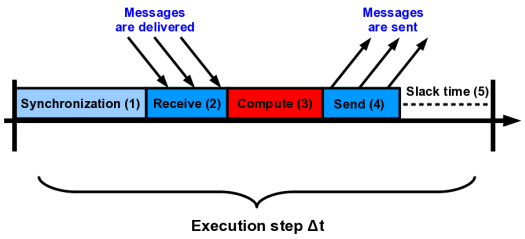

Figure 4: Periodic federate scheme

Historically, in DIS simulation standard (27), this synchronization phase is made (for each federate) by consulting global wall clock time (WCT) available for each simulator. ONERA and CNES works present a new original synchronization mechanism by sending an interaction from the fastest federate, called *pulse*, which rhythms the whole simulation run-time. In Fujimoto and McLean works, synchronization and reception phases are made in the same time by time management mechanisms. To summarize, the synchronization phase can be done either by three different methods:

1. Consult the hardware clock on a mono-processor system or use a distributed hardware clock like Real-Time Clock and Interrupt Module (RCIM)

system for distributed applications available on our Linux Red Hawk platforms;

2. The federate which has the highest speed cycle sends an interaction to all others in order to rhythm the execution of all others federates involved in the federation;

3. Use of Time Management HLA mechanisms to ensure messages delivery in all federation and synchronize every federates steps. Note that, these time steps could be different according to application requirements.

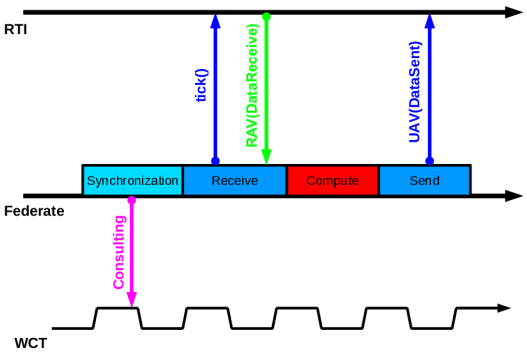

Figure 5: Data Flow execution mode

Execution modes

We distinguish two different run-time modes based on periodic federates. The first one is the *Data flow model*. This kind of execution mode is only scheduled by the communication flow between each federate. Each federate waits for a data to run its local algorithm and computes its own new data for the rest of the federation. This approach could only be used on synchronous distributed systems like PRISE Red Hawk RCIM synchronized nodes. Federates communicate using HLA basic *publish and subscribe* principles through RTI services calls like `updateAttributeValues()` (*Cf.* Figure 5). We assume that the receiver federate is waiting for a `reflectAttributesValues()` callback in reception phase. Each federate then runs its own algorithm when it receives an available input data. The main interest of this run-time execution mode is the simplicity of modeling its behavior with a formal model compliant with real-time scheduling policies and techniques. However, the developers have to ensure by programming which cycle receives which data. This approach is not very suitable for adding new federate or to plug existing federates to another federation execution. Most of all, there is no safety guarantee during the run time. If the application was not well scheduled, a federate could always be blocked (waiting for an expected data). So one needs to be accurate with the formal model and its implementation to ensure good execution of the whole federation.
Other execution mode use time-management mechanisms provided by HLA standard (28)(*Cf.* Figure 6). During the run-time, each federate computations and communications are scheduled by time management principles and algorithms. A suitable deployment of these techniques ensures a consistent temporal behaviour on a common time reference : the *simulated time axis*. This approach is the best way to maintain consistency between federates located on asynchronous computers (no common Wall Clock Time). The main advantage of time management is the possibility to easily add some new federates. The temporal behavior and consistency of the whole simulation is based on simulated time coherence. The time advance could also be correlated to an hardware clock to ensure the respect of real time constraints. Accordingly with the HLA standard, all federates are both regulators and constrained. Two kinds of services allow the federate to express its requests for advancing its local logical time:

- `nextEventRequest(`t`)` service (noted NER(t)): This service allows to receive the next event available for asked simulated time NER() and then a `timeAdvanceGrant(`t'`)` callback (noted TAG(t')) given by the RTI with a time stamp equal to the time stamp t' of the simulation message (t' could be less than t). This kind of federate is called *Event-Driven federates*;

- `timeAdvanceRequest(`t`)` service (noted TAR(t)): This service ensures the delivery of all available messages. The RTI grants this logical time advance (guaranteeing causality constraints) by invoking all the available `reflectAttributeValue()` callbacks (noted RAV()) and finally by accepting the time advance through the invocation of the `timeAdvanceGrant(`t`)` callback. This kind of federate is called *Time-Driven federates*.

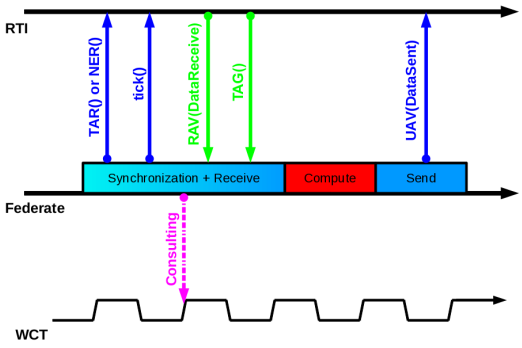

Figure 6: Time management execution mode

Necessity of formal proof for real-time

To our knowledge, no related work from simulation community has linked any formal model from scheduling theory with concepts of distributed simulations (especially with HLA standard). Thus real-time simulations are usualy validated by experiments rather than formal models and schedulability analysis. But, we claim that some formal models compliant with schedulability techniques are essential to validate real-time behavior of our simulations. For example, research on RT CORBA standard have investigate the validation of the global end to end behaviour by combining scheduling techniques Deadline Monotonic algorithm, DPCP[7] (29) and an algorithm to map priorities founded by formal results to local priorities provided by local operating systems on each node.

Figure 2 shows that each federate (each computation made by a federate) is illustrated by a box. Each CERTI communication between federate is represented by a an arrow. These data dependencies could be modelized by using differents techniques. In previous paper (30), we show the feasibility to formally validate basic Data Flow simulations on monoprocessor system by combining Deadline monotonic techniques and simple precedence constraints. We extend this formalism to describe our distributed Data Flow applications by using Tindell and Clark holistic method (31).

In order to add determinism to first generation time management technics involved in CERTI software (based on Chandy-Misra-Bryant algorithm (32)), we recently propose an analytical methodology to formally quantify the number of null messages exchanged between each time-driven real-time periodic federates (federates which use `timeAdvanceRequest()` service) involved in a real time simulation (33). We also add a new algorithm called NULL MESSAGE PRIME adapted to event-driven real-time periodic federates (federates which use `nextEventRequest()` service) which exhibits very interesting properties, including a solution to the time creep problem. We currently investigate some model checking by using UPPAAL tool (34) in order to exhibit formal proofs and have better evaluation of time management services and their implementation. The formal validation part of our works is not described in present paper, we here focusing on experimental aspects.

Experimentals results

WCET and WCTT measurements

The execution time of a program usually depends on the input data. In the context of real-time systems, it is necessary to be able to estimate the Worst-Case Execution Time or WCET. For hard real-time systems, it is necessary to assess the execution time in the worst case to properly size the system and find the best allocation of tasks among the processors. In our case, we have made some measurements of execution time for a given temporal complexity of algorithm (*Cf.* Table 1). We assume that spatial complexity (memory) is properly dimensionned according to embedded systems requirements.

$O(n^m)$	$m=1$ *(ms)*	$m=2$ *(ms)*	$m=3$ *(ms)*	$m=4$ *(ms)*
$n=10$	**0.001**	**0.003**	**0.007**	**0.065**
$n=20$	**0.001**	**0.005**	**0.052**	**1.182**
$n=30$	**0.001**	**0.008**	**0.181**	**5.838**
$n=40$	**0.001**	**0.010**	**0.386**	**18.240**
$n=50$	**0.001**	**0.015**	**0.803**	**44.077**

Table 1: Execution time of an algorithm with $O(n^m)$ complexity

Calculation Worst Case Transit Time values for all messages through CERTI must take into account three phases (*Cf.* Figure 7). Phase 1 is the copy on local host Unix domain socket and the local computation of the Sender Federate associated RTIA process. Phase 2 describes the time to read and write on different communication TCP (or UDP) sockets over the network or on the local host, and the time needed for RTIG local computation. Phase 3 is the copy on local host Unix domain socket and the local compute of Receiver Federate associated RTIA process.

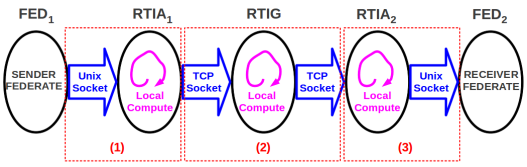

Figure 7: CERTI communication steps

We have developed a benchmark called PING_PONG used to measure CERTI communication latency. Two federates PING and PONG exchange messages (with a given size specified by user) through CERTI RTI. Table 2 gathers experimental measurements (given in milliseconds) of CERTI transit time with respect to three configurations:

- *Configuration 1*: Federate PING, federate PONG, both RTIAs and RTIG run on one single PRISE Red Hawk node;
- *Configuration 2*: Federate PING, its RTIA and RTIG run on a single PRISE Red Hawk node and Federate PONG and its RTIA run on another node;

[7] *Distributed Priority-Ceiling Protocol*

- *Configuration 3*: Federate PING, its RTIA run on a single PRISE Red Hawk node, Federate PONG and its RTIA run on another node and finally RTIG run also on its own PRISE node;

Message size	*Conf 1 (ms)*	*Conf 2 (ms)*	*Conf 3 (ms)*
100 bytes	**0.293**	**0.252**	**0.236**
500 bytes	**0.315**	**0.263**	**0.256**
1000 bytes	**0.353**	**0.281**	**0.286**
5000 bytes	**0.406**	**0.411**	**0.422**
10000 bytes	**0.422**	**0.478**	**0.522**
50000 bytes	**1.066**	**1.372**	**1.607**

Table 2: CERTI WCTT Measurements

Data Flow execution mode

As illustrated in Figure 2, federate 1 and federate 4 run a loop of 50 ms (20 Hz) and federate 2 and federate 3 run a loop of 10 ms (100 Hz). The data flow execution model have a good behavior for real-time purpose on our specific plat-form. Federate 1 and 4 compute an algorithm in $O(30^4)$ and Federate 2 and 3 compute an algorithm with complexity equal to $O(10^4)$. Table 3 show that all cycles respect corresponding period (10 ms and 50 ms) and the global behavior is stable for all cycles.

	Min	**Mean**	**Max**	**Std. Dev.**
Federate 1	**49.394**	**49.449**	**49.585**	**0.059**
Federate 2	**9.056**	**9.119**	**9.458**	**0.127**
Federate 3	**9.058**	**9.131**	**9.501**	**0.146**
Federate 4	**49.01**	**49.077**	**49.150**	**0.058**

Table 3: Federates Cycle Duration (Data Flow Periodic)

We also focus on the accleration of the application rhythm to allow the federation to run as fast as possible. For these experiments, we keep the speed ratios between the different federates cycles. Thus federates 1 and 4 are five times slower than federates 2 and 3 (and also corresponding communications). We retain the complexity of the algorithms computed by each federate. Table 4 show that, with corresponding algorithms complexities, faster federates (2 and 3) could respect a computational period equal ***2 ms*** and slower federates could ensure the respect of a period less than ***10 ms***. These results show that CERTI could ensure high frequency communicating processes with Data Flow execution mode.

	Min	**Mean**	**Max**	**Std. Dev.**
Federate 1	**6.108**	**6.136**	**6.292**	**0.0558**
Federate 2	**1.041**	**1.176**	**2.119**	**0.267**
Federate 3	**1.045**	**1.213**	**2.082**	**0.354**
Federate 4	**6.048**	**6.158**	**6.355**	**0.092**

Table 4: Federates Cycle Duration (Data Flow As Fast as Possible)

Time Management execution mode

For time management model, classical null message algorithm implemented in CERTI seems to have a good behavior to ensure real-time properties to our simulator (refer to table 5). Indeed, all computed cycles are respected (10 ms and 50ms); the global behavior is also very regular.

	Min	**Mean**	**Max**	**Std. Dev.**
Federate 1	**48.64**	**49.765**	**50.807**	**0.532**
Federate 2	**9.514**	**9.592**	**10.618**	**0.172**
Federate 3	**9.372**	**9.624**	**10.959**	**0.248**
Federate 4	**48.029**	**49.474**	**50.787**	**0.841**

Table 5: Federates Cycle Duration (Time Management Periodic)

One more time, we acclerate the application rhythm to allow the federation to run as fast as possible by using classical CERTI time management implementation. The use of `TAR()` (HLA services calls) for each federate steps seems to generate some overhead (compared with Data flow model). In this case, the number of NULL messages generated by original algorithm is acceptable for real-time specification (hard real time deadline). Table 6 show that, with corresponding algorithms complexities, faster federates (2 and 3) could respect a computational period equal ***6 ms*** and slower federates could ensure the respect of ***15 ms*** period. These results show that CERTI could ensure high frequency communicating processes as well with Time management execution. As a conclusion, time management mechanisms provided by CERTI middleware enforce a good synchronization for our kind of real-time federates.

	Min	**Mean**	**Max**	**Std. Dev.**
Federate 1	**13.266**	**13.376**	**13.607**	**0.100**
Federate 2	**1.582**	**2.676**	**6.487**	**1.883**
Federate 3	**1.544**	**2.678**	**6.587**	**1.875**
Federate 4	**13.293**	**13.427**	**13.766**	**0.139**

Table 6: Federates Cycle Duration (Time Management As Fast As Possible)

Perpectives and Conclusion

We propose, in this paper, experimental results from our work on real-time simulations with our CERTI middleware. However, real-time analysis required the modelization of several aspects of a distributed simulation. Different static scheduling and run time analysis have been studied under different hypothesis (single processor, distributed synchronous processors, distributed asynchronous processors, ...). Interested reader could refer to previous papers (30) (33) to get a more complete description of formal part of our work.
This paper shows that current CERTI performances are very good for real-time and/or high performance simulations. We have also develop and updated a lot of tools to manage the allocation of both federate and CERTI processes over PRISE processors and modify the priority of each one for compliance with scheduling technique used. These new implementations, that are not described in present paper, help to ensure better responsiveness of HLA services. Indeed, we pursue our efforts and we currently work on HP-CERTI approach (35) to replace Unix and TCP communication sockets through shared memories (for exchange on the same node). In addition, we will evaluate the use of multi-threading for process RTIG and ensure real-time properties for all messages passing through it. As well, we plan to use real-time dynamic memory allocators from TLSF library (36) and first experiments show promising results.
We have recently implemented and tested an HLA aircraft component-based federation composed by nine federates, each representing a specific part of the aircraft or environment (37). This simulation is human-in-the-loop and the operator could interact with the simulation by a federate which acquires the user orders transmitted by a real yoke/throttle/pedals system. Now, we think that our work on real-time simulations is mature (as well as our middleware CERTI). Indeed, hard real-time properties of our architecture (and both techniques to manage it) could allow the connection of simulators with real physical actuators and sensors and/or real embedded systems to run hardware-in-the-loop simulations with high-frequency requirements.

REFERENCES

[1] J.A. Stankovic, ***Misconceptions about real-time computing***, IEEE Computer Journal, 1988.

[2] Object Management Group, ***Minimum CORBA-Joint Revised***, OMG Document, orbos/98-08-04, 1998.

[3] J.A. Stankovic, ***Distributed real-time computing: the next generation***, Technical report, University of Massachussets Amherst, 1992.

[4] T. Abdelzaher, S. Dawson, W.C. Feng, F. Jahanian, S. Johnson, A. Mehra, T. Mitton, A. Shaickh, K. Shin, Z. Wang, H. Zou, ***ARMADA Middleware Suite***, Proceedings of the IEEE Workshop on Middleware for Distributed Real-Time Systems and Services, San Francisco, December 1997.

[5] O. Gonzalez, C. Shen, I. Mizunuma, M. Takegaki, ***Implementation and Performance of MidART***, Proceedings of the IEEE Workshop on Middleware for Distributed Real-Time Systems and Services, San Francisco, December 1997.

[6] Object Management Group, ***Real-time CORBA Specifications***, OMG Document formal/05-01-04, Version 1.2, 2005.

[7] Object Management Group, ***Data Distribution Service for Real-time Systems***, OMG Document formal/07-01-01, Version 1.3, 2007.

[8] The Institute of Electrical and Electronics Engineers (IEEE) Computer Society, ***IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules***, Simulation Interoperability Standards Committee, 2010.

[9] The Institute of Electrical and Electronics Engineers (IEEE) Computer Society, ***IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification***, Simulation Interoperability Standards Committee, 2010.

[10] The Institute of Electrical and Electronics Engineers (IEEE) Computer Society, ***IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification***, Simulation Interoperability Standards Committee, 2010.

[11] K.W.Arthur, K.S.Booth, ***Evaluating 3D Task Performance for Fish Tank Virtual Worlds***, ACM Trasactions of Information Systems, Vol. 11, N3, pages 239-265, July 1993.

[12] E.Noulard, B.D'Ausbourg, P.Siron, ***Running Real Time Distributed Simulations under Linux and CERTI***, European Simulation Interoperability Workshop, 2007.

[13] H.Zao, ***HLA Streaming and Real-Time Extensions***, Phd thesis, School of Information Technology Engineering, University of Ottawa, 2001.

[14] Airbus Avionics & Simulation Organisation, ***Avionics and Simulation Products***, EDYY presentation for engineer schools, May 2008.

[15] S. Bachinsky, J. Noseworthy, F.Hodum, ***Implementation of the Next Generation RTI***, Proceedings of the Spring Simulation Interoperability Workshop, Orlando, Florida, USA, 1999.

[16] T.McLean, R.Fujimoto, B.Fitzgibbons, ***Middleware for real-time distributed simulationsl***, Concurrency and Computation: Practice and Experience, 2004.

[17] H.Zao, N.D.Georganas, ***Architecture proposal for Real-Time RTI***, Proceedings of the Simulation Interoperability Standards Organization (SISO) Simulation Interoperability Workshop, 2000.

[18] A.Boukerche, L.Kaiyuan, ***A Novel Approach to Real-Time RTI Based Distributed Simulation System***, Proceedings of the 38th annual Symposium on Simulation, 2005.

[19] D.Bruzman, M.Zyda, K.Watsen, M.Macedonia, ***Virtual Reality Transfer Protocol design rational***, Proceedings of the sixth IEEE Workshop on Enabling Technologies, 1997.

[20] R.Jansen, W.Huiskamp, J.Boomgaardt, M. Brassé, ***Real-time Scheduling of HLA Simulator Components***, Euro Simulation Interoperability Workshop, 2004.

[21] Concurrent Computer Corporation, ***Real-Time Clock and Interrupt Module Users Guide***, User Guide, August 2001.

[22] J.Baietto, ***Real-Time linux: The RedHawk Approach***, Concurrent Computer Corporation, White Paper.

[23] B.O.Gallmeister, ***POSIX.4: programming for the real world***, O'Reilly & Associates, Inc. 1995.

[24] P. Siron, E. Noulard, J.-Y. Rousselot, ***CERTI : an open Source RTI, why and how***, Fall Simulation Interoperability Workshop, 2009.

[25] R.M.Fujimoto, ***Zero Lookahead And Repeatability In The High Level Architecture***, Proceedings of the 1997 Spring Simulation Interoperability Workshop, 1997.

[26] R.M.Fujimoto, T. McLean, ***Repeatability in real-time distributed simulation executions***, Proceedings of the fourteenth workshop on Parallel and distributed simulation, 2000.

[27] DIS Steering Commitee, ***The DIS Vision, A Map to future of Distributed Simulation***,Tech. Report from Institute for Simulation and Training, 1994.

[28] R.M.Fujimoto, ***Time Management in the High Level Architecture***, Simulation, 71, pp 388-400. December 1998.

[29] L,Dipippo, L.Cingiser, V.F.Wolfe, L.Esibov, G.B.Bethmangalkar, ***Scheduling and Priority Mapping for Static Real-Time Middleware***, Real-Time Systems, Vol 20, Kluwer Academic Publishers, 2001.

[30] J-B.Chaudron, P.Siron, M.Adelantado, ***Towards an HLA Run-time Infrastructure with Hard Real-time Capabilities***, Proceedings of the European Simulation Interoperability Workshop, 2010.

[31] K.Tindell, J.Clark, ***Holistic Schedulability Analysis for Distributed Hard real-time systems***, Microprocessing & Microprogramming, 1994.

[32] K.M.Chandy, J.Misra, ***Distributed Simulation: A Case Study in Design and Verification of Distributed Programs***, Software Engineering, IEEE Transactions, 1979.

[33] J-B.Chaudron, P.Siron, E.Noulard, ***Design and model-checking techniques applied to real-time RTI time management***, Proceedings of the Spring Simulation Interoperability Workshop, 2011.

[34] G.Behrmann, A.David, K.G.Larsen, ***A Tutorial on UPPAAL***, White Paper, Department of Computer Science, Aalborg University, Denmark, November 2004.

[35] M.Adelantado, J.L.Bussenot, Jean-Loup, J.Y.Rousselot, P.Siron, M.Betoule, Marc, ***HP-CERTI : Towards a high Performance, high Availability Open Source RTI for Composable Simulations***, Fall Simulation Interoperability Workshop, 2004.

[36] M.Masmano, I.Ripoll, A.Crespo, J. Rea, ***TLSF: A New Dynamic Memory Allocator for Real-Time Systems***, Real-Time Systems, Euromicro Conference on, 2004.

[37] J-B.Chaudron, D.Saussie, P.Siron, M. Adelantado, ***Real-time Aircraft Simulation using HLA standard: An overview***, Simulation in Aerospace Conference, Toulouse, June 2011.