HLA HIGH-PERFORMANCE AND REAL-TIME SIMULATION STUDIES WITH CERTI

Jean-Baptiste Chaudron, Martin Adelantado and Eric Noulard
Office National d'Etudes et de Recherches Aérosaptiales
2 Avenue Edouard Belin
FR-31055 Toulouse Cedex 4, France
E-mail: name.lastname@onera.fr

Pierre Siron
Institut Supérieur de l'Aéronautique et de l'Espace
10 Avenue Edouard Belin
FR-31055 Toulouse Cedex 4, France
E-mail: name.lastname@isae.fr

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

ABSTRACT

Our work takes place in the context of the HLA (High Level Architecture) 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 RTIs (Run Time Infrastructures). 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 requirements to experimental tests for validation of distributed real-time simulation with our own RTI called 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 (Stankovic 1988). 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 (Common Object Request Architecture) (OMG 2002) 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.

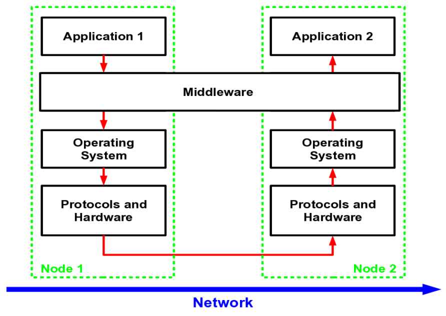

Figure 1: Illustration of Middleware

Indeed, real-time experts investigate distributed computing solutions to ensure real-time behavior for complex systems (Stankovic 1992). 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 particular research projects like ARMADA (Abdelzaher et al. 1997) and MIDART (MIDdleware Architecture for distributed Real-Time systems) (Gonzalez et al. 1997) and also some advances in current standards to include real-times properties, like RT CORBA (Real-Time CORBA) (OMG 2005) or more recently DDS (Data Distribution Service) (OMG 2007). The main objective of our work is to use an HLA middleware, compliant with current HLA IEEE 1516-2010 standard (IEEE 2010a) (IEEE 2010b) (IEEE 2010c) to develop, interconnect and maintain real-time simulations of embedded system (hardware-in-the-loop system or 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 (Plate-forme de Recherche et d'Ingénierie des Systèmes Embarqués). The main focus of this project is to study new embedded system concepts and techniques through a special hardware and software environment. 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 (Office National d'Etudes et de Recherches Aérosaptiales) and CNES (Centre National d'Etudes Spatiales) laboratories gave first elements to understand the use of the HLA standard and CERTI run-time infrastructure for real-time simulations (Noulard et al. 2008). 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 dynamics of the satellite 1; ***Federate 3*** is a simulator of the dynamics of the satellite 2 and finally ***Federate 4*** is a simulator of the board computer on satellite 2.

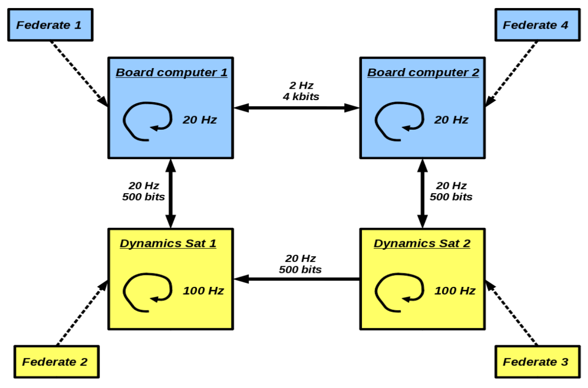

Figure 2: CNES Satellites formation flying simulation

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 (Zhao 2001). We claim that the choice of a distributed computing standard and its underlying middleware is an important starting point to 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. The RTI is the distributed software for interconnecting various federates to a global federation execution. The RTI-NG (RTI Next Generation) (Bachinsky et al. 1999) 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 (Zhao and Georgeanas 2001) (McLean et al. 2004) (Boukerche and Kaiyuan 2005) ;
2) Global scheduling services in RTI (Zhao and Georgeanas 2001) (Boukerche and Kaiyuan 2005) ;
3) Real-time Optimized RTI services like time Management from Fujimoto and McLean (McLean et al. 2004) or Data Distribution Management for Boukerche works (Boukerche and Kaiyuan 2005) ;
4) Quality of service communication with, for example, RSVP (Ressource ReSerVation Protocol) (Zhao 2001) or specific protocols like VRTP (Virtual Reality Transfer Protocol) (Brutzman et al. 1997) ;
5) Use a real-time operating system to allow preemptive priority scheduling (Jansen et al. 2004).

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 (Zhao 2001) (Zhao 2001) which used RTI-NG).

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 GPUs (Graphics Processing Units*)*, 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 (Concurrent Computer Corporation 2001).
Software: Linux Red Hawk (Baietto et al. 2008) Operating system compliant with POSIX real-time standard (Gallmeister 1995). This RTOS (Real Time Operating System) 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 concluded that this operating system is suitable for real-time computing (Jansen et al. 2004).
Middleware: In our approach, we will rely on our Open Source RTI called CERTI because we have a complete knowledge off its implementation.

CERTI Middleware

For years, the French Aerospace Laboratory (ONERA) has been developing his own Open-Source middleware RTI compliant with HLA standard called CERTI (Siron et al. 2009). This RTI runs on 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 RTIA (RTI Ambassador) process through a Unix-domain socket (equivalent to Local Run-time Component or LRC). The RTIA processes exchange messages over the network, in particular with the RTIG (RTI Gateway) process (equivalent to Central Run-time Component or CRC), via TCP (and also UDP) sockets, in order to run the distributed algorithms associated with the RTI services.

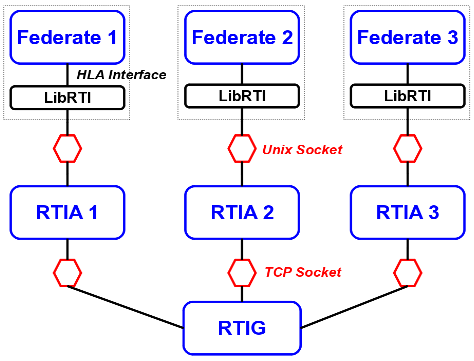

Figure 3: CERTI architecture

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 level. In our case, a key benefit is to master the implementation of RTI used and thus be able to incorporate changes in the source code to ensure temporal predictability of CERTI.

OUR APPROACH

Towards periodic federates

The concept of periodic federates, named "*repeatability within simulations*" has been introduced by Fujimoto and McLean (Fujimoto 1997) (Fujimoto and McLean 2000) 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 **Δt**. 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 (Noulard et al. 2008) show the necessity of explicitly adding a synchronization phase to ensure the global coherent run time of the whole simulation (*Cf.* Figure 4).

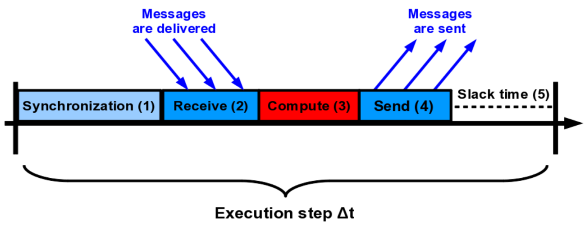

Figure 4: Periodic federate scheme

Historically, in DIS simulation standard (Cheung and Loper 1994), this synchronization phase is made (for each federate) by consulting global WCT (Wall Clock Time) 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 RCIM (Real-Time Clock and Interrupt Module) 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.

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.

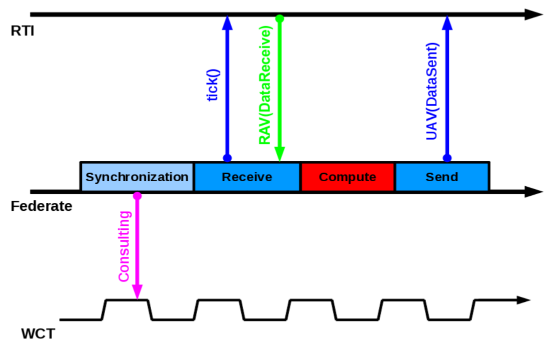

Figure 5: Data Flow execution mode

Other execution modes use time-management mechanisms provided by HLA standard (Fujimoto 1998). 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 behavior 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)*** and ***timeAdvanceRequest(t).***

The ***nextEventRequest(t)*** service (noted ***NER(t)***) 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 (*Cf.* Figure 6).

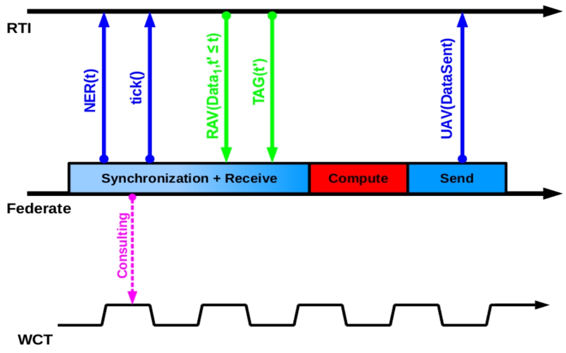

Figure 6: Event Driven execution mode

The ***timeAdvanceRequest(t)*** service (noted ***TAR(t)***) 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 Stepped* federates (*Cf.* Figure 7).

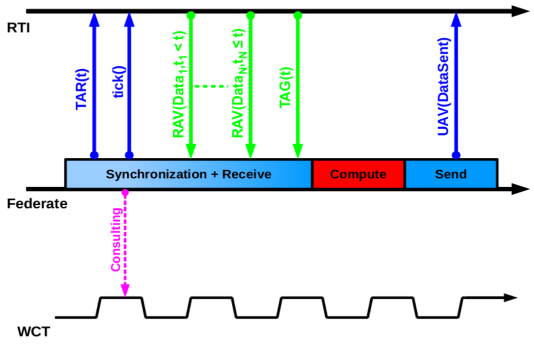

Figure 7: Time Stepped execution mode

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 usually 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, researches on RT CORBA standard have investigated the validation of the global end to end behavior by combining scheduling techniques Deadline Monotonic algorithm, DPCP (Distributed Priority-Ceiling Protocol) (Dipippo et al. 2001) 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 an arrow. These data dependencies could be modeled by using different techniques. In previous paper (Chaudron et al. 2010), we showed the feasibility to formally validate basic Data Flow simulations on mono-processor system by combining Deadline monotonic techniques and simple precedence constraints. We extend this formalism to describe our distributed Data Flow applications by using and adapting Tindell and Clark holistic method (Tindell and Clark 1994). These techniques allows to take into account the dependency between the scheduling of tasks and messages in distributed real-time systems.

In order to add determinism to first generation time management mechanisms involved in CERTI software (based on Chandy-Misra-Bryant algorithm (Chandy and Misra 1979)), 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 (Chaudron et al. 2011). 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 (Behrmann et al. 2004) 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 are 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 WCET (Worst-Case Execution Time). 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 dimensioned according to embedded systems requirements.

$O(n^m)$	*m=1*	*m=2*	*m=3*	*m=4*
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 (in milliseconds)

Calculation of WCTT (Worst Case Transit Time) values for any CERTI message must take into account three phases (*Cf.* Figure 8). 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.

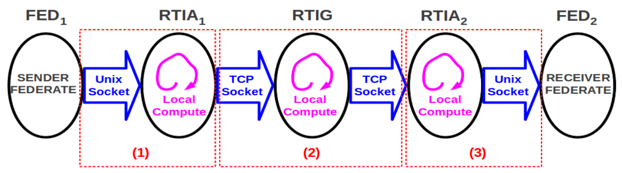

Figure 8: CERTI communication steps

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. 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;
- ***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	*Config. 1*	*Config. 2*	*Config. 3*
100 bits	0,293	0,252	0,236
500 bits	0,315	0,263	0,256
1000 bits	0,353	0,281	0,286
5000 bits	0,406	0,411	0,422
10000 bits	0,422	0,478	0,522
50000 bits	1,066	1,372	1,607

Table 2: CERTI WCTT measurements (in milliseconds)

Data Flow execution mode

For CNES federation, 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 has 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 shows that all cycles respect corresponding periods (10 ms and 50 ms) and the global behavior is stable for all cycles.

	Min	*Mean*	*Max*	*Std. Dev.*
Fed. 1	49,394	49,449	49,585	0,059
Fed. 2	9,056	9,119	9,458	0,127
Fed. 3	9,058	9,131	9,501	0,146
Fed. 4	49,010	49,077	49,150	0,056

Table 3: Federate cycle duration in milliseconds (Data Flow periodic)

We also focus on the acceleration 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.*
Fed. 1	6,108	6,136	6,292	0,056
Fed. 2	1,041	1,176	2,119	0,267
Fed. 3	1,045	1,213	2,082	0,354
Fed. 4	6,048	6,158	6,355	0,092

Table 4: Federate cycle duration in milliseconds (Data Flow as fast as possible)

Time Management execution mode

For time management model, we choose the Time Stepped execution mode to ensure consistency between the real time and the simulated time. In this case, classical null message algorithm implemented in CERTI seems to have a good behavior to ensure real-time properties to our simulator (*Cf.* table 5). Indeed, all computed cycles are respected (10 ms and 50ms); the global behavior is also very regular even if some irregularities appears compared to Data Flow execution mode.

	Min	*Mean*	*Max*	*Std. Dev.*
Fed. 1	48,640	49,765	50,807	0,532
Fed. 2	9,514	9,592	10,618	0,172
Fed. 3	9,372	9,624	10,959	0,248
Fed. 4	48,029	49,474	50,787	0,841

Table 5: Federate cycle duration in milliseconds (Time Management periodic)

One more time, we accelerate 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 shows that, with corresponding algorithms complexities, faster federates (2 and 3) could respect a computational period equal ***7 ms*** and slower federates could ensure the respect of ***15 ms*** period (for the worst case). 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.*
Fed. 1	13,266	13,376	13,607	0,100
Fed. 2	1,582	2,676	6,487	1,883
Fed. 3	1,544	2,678	6,587	1,875
Fed. 4	13,293	13,427	13,766	0,139

Table 6: Federate cycle duration in milliseconds (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 requires the modeling 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 (Chaudron et al. 2010) (Chaudron et al. 2011) 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 developed 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 (High-Performance CERTI) approach (Adelantado et al. 2004) 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 (Two-Level Segregate Fit) library (Masmano et al. 2004) 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 (article under submission process). 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

Abdelzaher, T.; Dawson, S.; Feng, W.C.; Jahanian,F.; Johnson,S.; Mehra, A.; Mitton, T.; Shaickh, A.; Shin, K.; Wang, Z. and Zou, H. 1997. “ARMADA Middleware Suite”. In *Proceedings of the IEEE RTSS 1997 Workshop on Middleware for Distributed Real-Time Systems and Services* (San Francisco, CA, USA, Dec).

Adelantado, M.; Bussenot, J.-L.; Rousselot, J.-Y.; Siron, P. and Betoule, M. 2004. “HP-CERTI: Towards a High Performance High Availability Open Source RTI for Composable Simulations”. In *Fall Simulation Interoperability Workshop FSIW-2004* (Orlando, Florida, USA, Sep 19-24).

Arthur, K.W. and Booth, K.S. 1993. “Evaluating 3D Task Performance for Fish Tank Virtual Worlds.” *ACM Transactions on Information Systems* 11, No.3 (Jul), 239-265.

Bachinsky, S.; Noseworthy, J. and Hodum, F. 1999. “Implemenation of the Next Generation RTI”. In *Spring Simulation Interoperability Workshop SSIW-1999* (Orlando, Florida, USA, Mar 14-19).

Baietto, J.; Korty, J.; Blackwood, J. and Houston, J. 2008. "Real-Time Linux : The Red Hawk Approach." Concurrent Computer Corporation White Paper (Sep).

Behrmann, G.; David, A. and Larsen, K.G. 2004. “A Tutorial on UPPAAL”. In *Proceedings of of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems SFM-RT '04.* (Cambridge, MA, USA, Jun 18-20), 179-186.

Boukerche, T. and Kaiyuan, L. 2005. “A novel Approach to Real-Time RTI Based Distributed Simulation System”. In *Proceedings of the 38th annual Symposium on Simulation ANSS '05* (San Diego, CA, USA, Apr 3-7).

Brutzman, D.; Zyda, M.; Watsen, K. and Macedonia, M. 1997. “Virtual Reality Transfer Protocol (VRTP) Design Rationale”. In *Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises WET-ICE '97* (Cambridge, MA, USA, Jun 18-20), 179-186.

Chandy, K.M. and Misra, J. 1979. “Distributed Simulation : A case study in design and verification of ditributed programs.” *IEEE Transactions on Software Engineering Journal 5*, No.5 (Sep), 440-452.

Chaudron, J.-B.; Siron, P. and Adelantado, M. 2010. “Towards an HLA Run-Time Infrastructure with Hard Real-Time Capabilities”. In *European Simulation Interoperability Workshop ESIW-2010* (Ottawa, Canada, Jul 12-14).

Chaudron, J.-B.; Siron, P. and Noulard, E. 2011. “Design and Model-Checking techniques applied to Real-Time RTI Time Management”. In *Spring Simulation Interoperability Workshop SSIW-2011* (Boston, MA, USA, Apr 4-8).

Cheung, S. and Loper, M. 1994. “Synchronizing Simulations in Distributed Interactive Simulation”. In *Proceedings of the 26th Winter Simulation Conference WSC '94* (Orlando, Florida, USA, Dec 11-14).

Concurrent Computer Corporation. 2001. "Real-Time Clock and Interrupt Module (RCIM) User's Guide." Technical Report 0891082-010 (Aug).

Dipippo, L.; Cingiser, L.; Wolfe, V.F.; Esibov, L.; Bethmangalkar, R.; Cooper, G.; Johnston, R.; Thuraisingham, B. and Mauer, J. 2001. “Scheduling and Priority Mapping for Static Real-Time Middleware.” *Real-Time Systems - Special issue on challenges in design and implementation of middlewares for real time systems* - 20, No.2 (Mar), 155-182.

Fujimoto, R. 1997. “Zero Lookahead and Repeatability in the High Level Architecture”. In *Spring Simulation Interoperability Workshop SSIW-1997* (Orlando, Florida, USA, Mar 3-7).

Fujimoto, R. 1998. “Time Management in the High Level Architecture.” Simulation *Journal* 71, No.6 (Dec), 388-400.

Fujimoto, R. and McLean, T. 2000. “Repeatability in Real-Time distributed simulation executions”. In *Proceedings of the fourteenth workshop on PArallel and Distributed Simulation PADS '00* (Bologna, Italie, May 28-31).

Gallmeister, B. 1995. *POSIX.4 Programmers Guide: Programming for the real world.* O'Reilly & Associates, Inc. Sebastopol, CA, USA (570p).

Gonzalez, O.; Shen, S.; Mizunuma, A. and Takegaki, M. 1997. “Implementation and Performance of MidART”. In *Proceedings of the IEEE RTSS 1997 Workshop on Middleware for Distributed Real-Time Systems and Services* (San Francisco, CA, USA, Dec).

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

IEEE – The Institute of Electrical and Electronics Engineers Computer Society. 2010. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules.* Simulation Interoperability Standards Commitee (Aug).

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

Jansen, R.; Huiskamp, W.; Boomgaardt, J. and Brassé, M. 2004. “Real-Time Scheduling of HLA Simulator Components”. In *European Simulation Interoperability Workshop ESIW-2004* (Edimburgh, Scotland, Jun 28-Jul 1).

Masmano, M.; Ripoll, I.; Crespo, A. and Real, J. 2004. “TLSF: A New Dynamic Memory Allocator for Real-Time Systems”. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems ECRTS '04.* (Catania, Italy, Jun 30-Jul 4), 79-86.

McLean, T.; Fujimoto, R. and Fitzgibbons, B. 2004. “Middleware for real-time distributed simulations.” *Concurrency and Computation: Practice & Experience - Distributed Simulation and Real-Time Applications Journal* 11, No.15 (Dec), 1483-1501.

Noulard, E.; D'Ausbourg, B. and Siron, P. 2008. “Running real time distributed simulations under Linux and CERTI”. In *European Simulation Interoperability Workshop ESIW-2008* (Edimburgh, Scotland, Jun 16-19).

OMG - Object Management Group. 2002. *Minimum CORBA Specification.* OMG Document formal/02-08-01 edition (Aug).

OMG - Object Management Group. 2005. *Real-Time CORBA Specification.* OMG Document formal/05-01-04 edition (Jan).

OMG - Object Management Group. 2007. *Data Distribution Service for Real-Time Systems.* OMG Document formal/07-01-01 edition (Jan).

Siron, P.; Noulard, E. and Rousselot, J.-Y. 2009. “CERTI : An Open-Source RTI, Why and How ?”. In *Fall Simulation Interoperability Workshop FSIW-2009* (Orlando, Florida, USA, Sep 21-25).

Stankovic, J.A. 1988. “Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems.” *IEEE Computer Journal* 21, No.10 (Oct), 10-19.

Stankovic, J.A. 1992. “Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems.” Invited paper for *Journal of the Society of Instrument and Control Engineers of Japan*.

Tindell, K. and Clark, J. 1994. “Holistic Schedulability Analysis for Distributed Real-Time Sytems.” M*icroprocessing & Microprogramming Journal 40*, 117-132.

Zhao H. 2001. “HLA Streaming and Real-Time Extensions”. *Phd Thesis*. School of Information Technology Engineering, University of Ottawa, Ontario, Canada. (Dec).

Zhao, H. and Georgeanas, N.D. 2001. “Architecture Proposal for Real-Time RTI”. In *Fall Simulation Interoperability Workshop FSIW-2001* (Orlando, Florida, USA, Sep 9-14).