This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 6844

To cite this document: Gervais, Clément and Chaudron, Jean-Baptiste and Siron, Pierre and Leconte, Régine and Saussié, David *Real-Time Distributed Aircraft Simulation through HLA.* (2012) In: 16th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications DS-RT 2012, 25-27 Oct 2012, Dublin, Ireland.

Real-Time Distributed Aircraft Simulation through HLA

Clément Gervais, Jean-Baptiste Chaudron,
Pierre Siron, Régine Leconte
Mathematics, Computer Science and Control Theory Department
Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)
Toulouse, France
Email: pierre.siron@isae.fr

David Saussié
Electrical Engineering Department
École Polytechnique de Montréal
Montréal,Canada
Email: david.saussie@polymtl.ca

***Abstract*—This paper presents some ongoing researches carried out in the context of the PRISE (Research Platform for Embedded Systems Engineering) Project. This platform has been designed to evaluate and validate new embedded system concepts and techniques through a special hardware and software environment. Since many actual embedded equipments are not available, their corresponding behavior is simulated using the HLA architecture, an IEEE standard for distributed simulation, and a Run-time infrastructure called CERTI and developed at ONERA. HLA is currently largely used in many simulation applications, but the limited performances of the RTIs raises doubts over the feasibility of HLA federations with real-time requirements. This paper addresses the problem of achieving real-time performances with HLA. Several experiments are discussed using well-known aircraft simulators such as the Microsoft Flight Simulator, FlightGear, and X-plane connected with the CERTI Run-time Infrastructure. The added value of these activities is to demonstrate that according to a set of innovative solutions, HLA is well suited to achieve hard real time constraints.**

***Keywords*-real-time HLA; aircraft simulation; CERTI.**

I. INTRODUCTION

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 [1] allows to consistently facing these problems. The term middleware describes a software agent operating as an intermediary between distributed processes. 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.

Modern flight simulation techniques and implementations often result in many sophisticated and complex calculations that require a high level of computing power. Several flight simulator applications often require their services to be delivered with respect to a given instant of time (deadline). This issue constitutes the problematic of real-time systems which are defined as 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 [2]. 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 plant control. Telephone switching system and image processing applications are examples of soft real-time systems. Figure 3 shows that our application is concerned by both types of real-time system characteristics.

Traditional standards and middleware architectures are not suitable for supporting real-time constraints. Real-time aircraft software and hardware components interconnected with middleware such as CORBA [3] have led to advances in current standards to include real-times properties, like Real-time CORBA [4] or more recently DDS [5]. The main objective of our work is to use the HLA IEEE 1516-2000 standard [6]–[8] , to develop, interconnect and maintain a flight simulator. However, works to include real-time specifications and properties to HLA standard are less advanced than others ones [9]. This article explains how we proceed to implement and test this simulator and how we validate real time behavior on our computing platform. The use of a distributed simulation architecture to study distributed embedded systems and hardware should provide a more natural and flexible framework for new researches in the domain.

II. BACKGROUND

Simulation is a well established technique used in the man-machine system area for training, evaluation of performance and research. Flight simulation re-creates how an airplane flies in its environment; it models the dynamic behavior of the flight vehicle under the action of aerodynamic, thrust and gravity forces, accordingly to the external environment characteristics (air density, wind, turbulence...).

To achieve this goal, a flight simulator consists of different components. The essential one still remains the mathematical description of the aircraft and its environment; the more accurate the model, the more realistic and reliable the simulation will be. Then a digital computer running a real-time operating system computes this model. The simulation can finally be completed with input organs (e.g. yoke-pedal systems, joysticks), display screens, cockpit-like environment and mechanical devices reproducing the aircraft motion (e.g. Stewart platform).

We claim that the choice of a distributed standard and its underlying middleware is an important requirement to obtain a high fidelity, valid and scalable real-time flight simulation. 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 but the different models are proprietary (and sometimes certified) as well as the Run Time Infrastructure. Indeed some works focus on the DDS standard for our flight simulator basis [10]. Other ones, like the authors of the present paper, choose the HLA standard [11]. The RTI (HLA underlying middleware) is the distributed software used for interconnecting various federates to a global federation execution. In [11], the authors use the RTI-NG [12] which was the first RTI 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. These works include optimized time management services [13], multi-threaded synchronous processes for RTI [13]–[15] and global scheduling services [14], [15]. These different techniques allow an improved use of system resources, better scalability and also a higher reactivity of services provided by the RTI.

III. HARDWARE AND SOFTWARE COMPONENTS

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.

A. Description

Our platform is built around the following components:

1) 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).
2) Software: Linux Red Hawk [16] Operating system compliant with POSIX Real time standard [17]. 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 [18].
3) A distributed clock technology allowing distributing same clock reference to each node [19].

B. CERTI

For years, the French Aerospace Laboratory (ONERA) has been developing his own Open-Source middleware RTI compliant with HLA standard called CERTI [20], running under several operating systems including Linux and Windows. We will use this RTI for interconnecting each component of the simulator. This RTI is recognizable through its original architecture of communicating processes. It is a distributed system involving two processes, a local one (RTIA) and a global one (RTIG), as well as a library (libRTI) linked with each federate. The CERTI architecture is depicted in Fig. 1. Each federate process interacts locally with an RTI Ambassador process (RTIA) through a Unix-domain socket. The RTIA processes exchange messages over the network, in particular with the RTIG process, via TCP (and also UDP) sockets, in order to run the distributed algorithms associated with the RTI services.

In our case, a key benefit of this architecture is to master the implementation of the used RTI and thus to facilitate the integration of changes in the source code to ensure temporal predictability of CERTI. Initial results, providing some answers about the suitability of CERTI to face real-time constraints, came from ONERA/CNES satellites formation flying studies [21]. These studies have shown that CERTI (in its original version) is able to manage multiple real-time federates with short period of time.

IV. SIMULATION ARCHITECTURE

A. Global view

The PRISE HLA Federation is composed of 11 federates, each representing a specific part of the aircraft or

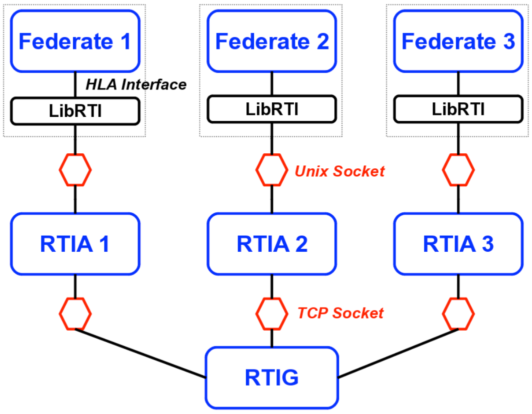

Figure 1. CERTI Architecture

environment (Fig. 2). New federates will be incorporated accordingly to the evolution of the requirements. For now, the federates go as follows:

- **Federate 0: Main controller** monitors and controls all simulation attributes;
- **Federate 1: Joystick** relays pilot inputs coming from the yoke/pedals device;
- **Federate 2: Cockpit** emulates user interface for autopilot settings via touchscreen;
- **Federate 3: EFCS** (Electronic Flight Control System) englobes all flight controllers and automatic pilot functions;
- **Federate 4 Control surfaces** simulates five different control surfaces (left/right ailerons, left/right elevators, rudder)
- **Federate 5 Engines** simulates two CFM56-5A1 turbofans;
- **Federate 6 Flight Dynamics** simulates the flight mechanics equations;
- **Federate 7 Sensors** simulates twenty different sensors of various kinds;
- **Federate 8 Environment** represents the US standard atmosphere (1976) and different turbulences/winds (Dryden, Von Karman, windshear) that could occur during the flight;
- **Federate 9 PFD** (Primary Flight Display) is a cockpit view with flight instruments;
- **Federate 10 Visualization** shows the aircraft in a virtual environment (FlightGear visualization engine, but as well Microsoft Flight Simulator or X-Plane).

B. *Real-time constraints*

The application is divided into two parts:

- the first part is concerned by hard real-time constraints and has to ensure that all deadlines are met for each federate. The Joystick and EFCS federates work at a frequency of 50 Hz, corresponding to an average frequency of the usual avionics system. The other federates (Engines, Control Surfaces, Flight Dynamics and Sensors) work at a frequency of 100 Hz; they simulate continuous-time systems modeled by differential equations and solved by numerical methods. Empiric experiments showed that 100 Hz was sufficient to deliver reliable results. Moreover, by taking too high a frequency, the real-time constraints would no longer be held.
- the second part deals with soft real-time constraints; the goal is to meet a certain subset of deadlines in order to optimize some application specific criteria. In our case, the Visualization and PFD federates work at a mean refresh rate of 60Hz in order to be fluid for human eyes [22].

V. TOWARDS A REAL-TIME EXECUTION

We present here the key steps that led us to a real-time execution of the federation.

A. *Run-time Execution Characteristics*

The calculation of Worst Case Execution Time (WCET) is a key issue for successfully schedule processes because it allows determining the C_i parameter value for a task. Calculation of the WCET should take into account specific calculations made by the federate. WCET were estimated for each federate. Moreover, Worst Case Transit Time (WCTT) were calculated for each message through CERTI middleware.

Current CERTI version does not provide any service or mechanism to ensure a real-time behavior of a simulation (federation). To manage every part of federation and to be compliant with formal techniques and scheduling techniques, different methods were added to the CERTI API (for Linux Operating System). The different implemented services also ensure a correct predictability for CERTI communications (WCTT) and federate computation (WCET).

We first implemented functions in CERTI that allow using affinity mechanism. CPU affinity is a scheduler property that assigns a process (federate, RTIA or RTIG) to a given set of CPUs on the system. The Linux scheduler will honor the given CPU affinity and the process will not run on any other CPU.

Another interface allows now the management of priority and scheduler for CERTI processes (including federates, RTIAs and RTIG). Modification of priority relies then on the choice of real-time scheduling algorithms under POSIX/Linux: two real-time algorithms, SCHED_FIFO and SCHED_RR, are intended for time-critical applications that need accurate control over the way in which runnable processes are selected for execution. Finally, we also use the mlockall mechanism for each federate, their respective RTIA and the RTIG processes in order to disable memory paging into the address space of the calling process.

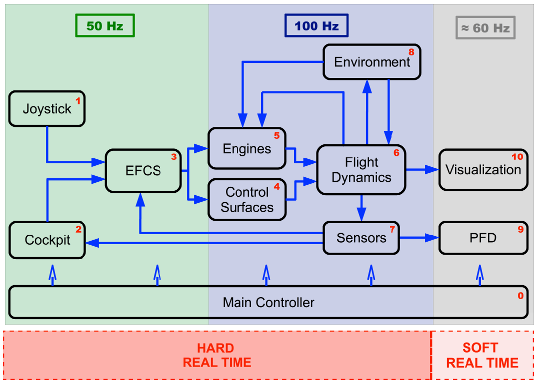

Figure 2. Federation description

B. Run time execution mode

Different executions modes were designed and implemented, each based on different approaches (data flow, time management). Depending on the chosen approach, a formal validation of the real-time behavior could be exhibited [23].

VI. CONCLUSION

The first but complete step of PRISE project has required the mastering of many aspects: from the realistic implementation of avionics code and environment models to the extension of HLA and CERTI distributed simulation to real-time. It will be easy to change the version of an existing federate by another (if its WCET is compliant with the global approach) in this modular and flexible architecture. It will also be easy to add new federates with an evolution of the global object model of the federation (HLA FOM) and a new real time analysis. We hope also that the defined FOM becomes a reference FOM for this research domain. We plan to study the flight formation of many systems by the duplication of the existing federates and the addition of the turbulence interaction with the addition of a new specialized federate.

REFERENCES

[1] Object Management Group, "Minimum CORBA - Joint Revised," August 1998.

[2] J. A. Stankovic, "Misconceptions about real-time computing," *IEEE Computer*, vol. 21, no. 10, 1988.

[3] T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The design and performance of a real-time corba event service," *SIGPLAN Notification*, vol. 32, no. 10, pp. 184–200, 1997.

[4] Object Management Group, "Real-Time CORBA Spécifications 1.2," Janvier 2005.

[5] ——, "Data Distribution Service for Real-time Systems Specification version 1.2," January, 2007.

[6] 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," September 2000.

[7] ——, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification," September 2000.

[8] ——, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification," September 2000.

[9] H. Zhao, "HLA Streaming and Real-Time Extension," Ph.D. dissertation, School of Information Technology Engineering, University of Ottawa, 2001.

[10] S. Zheng, J. He, J. Jin, and J. Han, "Dds based high fidelity flight simulator," in *Proceedings of the 2009 WASE International Conference on Information Engineering*, 2009.

[11] A. Lemmers, P. Kuiper, and R. Verhage, "Performance of a component-based flight-simulator architecture using the hla paradigm," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, ser. AIAA-2002-4476, August 2002.

[12] F. Hodum, J. Noseworthy, and S. Bachinsky, "Implementation of the next generation rti," *Spring Simulation Interoperability Workshop*, 1997.

[13] T. McLean, R. Fujimoto, and B. Fitzgibbons, "Middleware for real-time distributed simulations," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 15, pp. 1483 –1501, 2004.

[14] H. Zhao and N. Georganas, "Architecture proposal for Real-Time RTI," *Proceedings of the Simulation Interoperability Standards Organization (SISO) Simulation Interoperability Workshop*, 2000.

[15] A. Boukerche and K. Lu, "A novel approach to real-time rti based distributed simulation system," in *Annual Simulation Symposium (ANSS) : Proceedings of the 38th annual Symposium on Simulation*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 267–274.

[16] J. Baietto, "Real Time Linux: The RedHawk Approach," *Concurrent Computer Corporation White Paper*, 2002.

[17] W. M. Corwin, D. C. Locke, and K. D. Gordon, "Overview of the ieee posix p1003.4 realtime extension to posix," *IEEE Real-Time Syst. Newsl.*, vol. 6, pp. 9–18, March 1990.

[18] Jansen, Roger and Huiskamp, Wim and Boomgaardt, Jan-Jell and Brassé, Marco, "Real-Time Scheduling of HLA Simulator Components," *Proceedings of the Simulation Interoperability Standards Organization (SISO) Spring Simulation Interoperability Workshop*, 2004.

[19] C. C. Corporation, "Real Time Clock and Interrupt Moule Users Guide," *User Guide*, August 2001.

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

[21] E. Noulard, B. d'Ausbourg, and P. Siron, "Running Real Time Distributed Simulations under Linux and CERTI," 2007.

[22] K. W. Arthur, K. S. Booth, and C. Ware, "Evaluating 3d task performance for fish tank virtual worlds," *ACM Transactions on Information Systems*, vol. 11, pp. 239–265, July 1993.

[23] J.-B. Chaudron, P. Siron, M. Adelantado, and E. Noulard, "HLA high performance and real-time simulation studies with CERTI," Guimaraes, Portugal, 2011.