Analysis and Optimization of time-management services in CERTI 4.0

Jean-Baptiste Chaudron
ISAE-SUPAERO, Université de Toulouse, France
jean-baptiste.chaudron@isae-supaero.fr

Pierre Siron
ISAE-SUPAERO, Université de Toulouse, France
pierre.siron@isae-supaero.fr

Martin Adelantado
ONERA, Toulouse, France
martin.adelantado@onera.fr

Keywords:
High-Level Architecture, Time-Management services, Run-Time Infrastructure, Discrete Event Synchronization

ABSTRACT: *Time management services are one of the key features of the High Level Architecture (HLA) IEEE simulation standard. Several algorithms allow the implementation of HLA time management services ranging from the Chandy-Misra-Bryant (CMB) null message conservative algorithm up to optimistic Jefferson time warp one. For years now, we are interested in enhancing the high and/or real-time performance of our open source RTI (CERTI). In this paper, we precisely describe our updated conservative time management algorithm which, under some assumptions, limits the time creep problem inherent to the classical CMB algorithm. We will provide detailed analysis and experimental results for different implementations of HLA time management services using our new release version called CERTI 4.0.*

1. Introduction

In the context of the PRISE project[1] (Research Platform for Embedded Systems Engineering) in ISAE-SUPAERO, we intend to provide to our students and researchers with a platform for the study, evaluation and validation of new embedded system concepts, architectures and techniques. For the design, prototyping and improvement of such concepts and systems, simulation tools and techniques are well-known, widely used and accepted among research institutes and industries. The simulation environment can be extremely complex and therefore require a lot of computing resources. Distributed computing paradigm proposes a high performance solution thanks to advances in network technologies where different programs located on several computers interact to achieve a global common goal. Designers and developers of distributed simulation applications had to face several problems such as heterogeneity of the various hardware and softwares components involved. The High Level Architecture (HLA) standard tackle this problem by providing a well known and documented framework for interoperability and re-usability of heterogeneous distributed simulation. It has been originally designed per the US Department of Defense (DoD) in 1996 and, nowadays, the current standard version is the IEEE 1516 Evolved [1], [2], [3].

The HLA standard provides multiple services to handle the different complex aspects of a simulation. Especially, the so-called *time management services* provided by HLA are one of the main benefits of this simulation standard [4]. These services allow to maintain a consistent global logical time throughout the whole simulation components by using different methods and algorithms. Specifically, each simulation message is assigned a logical time-stamp, and the use of these services ensures that messages are delivered to each simulator (*i.e* federate) in the logical time-stamp order, and no message is delivered to a simulator in its past (with respect to the logical time). Two types of approaches, which ensure the causality constraint, have been proposed in the literature. The first one is the optimistic strategy where each message is processed by the simulator in order of their arrival until it detects a logical timing violation (*i.e* a message in the past) and requires a mechanism for turning back (roll-back mechanism) [5]. The second one is the conservative strategy which avoids the violation of the local causality constraint altogether during run-time. The HLA standard does not provide or advise a specific implementation for this Time Management service offered by the RTI. Based on our knowledge about HLA, its usage and application, we are always working

[1]French acronym for *Plate-forme pour la Recherche en Ingnierie des Systmes Embarqus*.

on extensions and improvement of our HLA Run-Time Infrastructure (RTI) implementation called CERTI. This paper focuses on the study of time management services implemented in CERTI and their optimization, the remainder of this paper is as follow:

- Section 2 recalls the history of the CERTI project, its time-management services and the evolutions of CERTI 4.0;
- Section 3 describes the different optimizations we implemented and investigated for time-management services;
- Section 4 presents our new open-source project used for this analysis and illustrates its usage per different results;
- Finally, a discussion of results, as well as currently planned extensions of the infrastructure, is proposed in Section 5.

2. An Open Source RTI named CERTI

2.1 Project History

CERTI is the name of an HLA-RTI. The CERTI project started in 1996 [6] in the early days of the HLA standard development. An open-source version was released in 2002 [7] and a major revision was done in 2009 [8]. Note, that reference [8] explains well why we have chosen to do an open-source version. The first objectives of ONERA were to understand and to master the HLA services semantic and the corresponding RTI implementation for researches in the field of distributed simulation for Aeronautics, Space and Defense, this RTI will be also useful for teaching and training and for industrial applications. The main contributors of this project are today ONERA and ISAE-SUPAERO, Université de Toulouse, France.

The list of conducted or on-going research projects is interesting. We give only some examples with the main keyword and the main reference, without detailing these projects and their related works:

- Security [9];
- Multi-Resolution [10];
- Bridges between HLA Federations [11];
- High-Performance [12][13];
- Time management analysis for real-time simulations [14][15];
- Scheduling HLA simulations [16];
- Real-Time and Aircraft simulation (collaboration with Ecole Polytechnique de Montréal) [17];
- Ptolemy-HLA (collaboration with University of California, Berkeley) [18] and, in particular, the relation between HLA time management and time notion in Ptolemy [19].

Many students have been and are involved in these research projects. They have also studied and demonstrated the interoperability between different flight simulators using CERTI (using different plugins): FlightGear (`http://www.flightgear.org/`), Xplane (`http://www.x-plane.com/`) and our flight simulator. The interoperability is illustrated by in formation flying. Regarding other teaching aspects, CERTI is used for practical works in different courses of ISAE-SUPAERO, Université de Toulouse: distributed systems, simulation for system engineering, validation of systems. A specific training for experimented users is given: HLA distributed modeling and simulation of complex systems.

2.2 Architecture description

As illustrated in Fig.1, CERTI is recognizable through its original architecture of communicating processes. CERTI architecture includes a local RTI Component (RTIA) for each federate and a central/global one (RTIG), as well as a library (libRTI) linked with each federate. Each federate process interacts locally with a RTI Ambassador process (RTIA) through a Unix-domain socket (or TCP socket on the Windows platform). The RTIA processes exchange messages over the network through the RTIG process, via TCP (and also UDP) sockets, in order to run the various distributed algorithms associated with the RTI services. The RTIG is the central gateway responsible for the delivery and the broadcast of relevant messages to all RTIA processes (depending on publication subscription of each federate). We will see in the next paragraph, as an illustration, how a classical time management algorithm is deployed on this architecture, and in the next chapter how we can take advantage of the central RTIG process for developing a new time conservative algorithm. The main programming language is C++, the libRTI has been written in the C++ or the Java language, different versions of this libRTI coexist corresponding to different versions of the HLA standard: 1.3, IEEE 1516-2000 or IEEE 1516-2010. The RTI also supports Python bindings and some bridges are maintained for Matlab and Fortran allowing a wide range of heterogeneous applications.

2.3 Time Management implementation in CERTI

The first generation of time management services are based on the so-called NULL Message Algorithm (NMA) from Chandy and Misra [20]. This is the main algorithm implemented in CERTI and it is used to avoid deadlock in a conservative federation. This approach is based on a contract for each federate called *lookahead*. Each federate undertakes not to send simulation messages with a logical timestamp less than its local time plus its lookahead. The respect of this contract enables the exchange

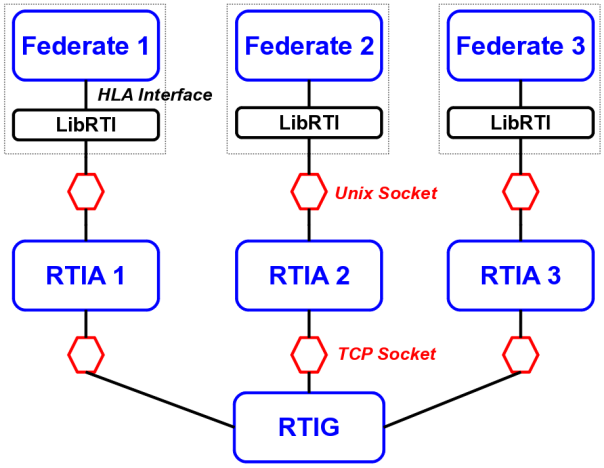

Figure 1. CERTI Architecture illustration

of additional messages called NULL messages (messages containing only time-stamps) indicating the Lower Bound on the Time Stamp[2] (LBTS) of future messages it could send. We will not detail this well known algorithm but we will try to resume the use of this algorithm with the HLA services and with the CERTI architecture. The time advancing phase begins with a `timeAdvancementRequest()` call (TAR) for a time-stepped federate or a `nextMessageRequest()` call (NMR) for a event-driven federate. It finishes by a `timeAdvanceGrant()` (TAG) callback (Illustrations are given Fig. 2 and Fig.3). All the federates are time regulating and constrained, the lookahead value is 1. The RTIA of the first federate receives a TAR or NER message and sends a NULL Message (NM) to the RTIG. The role of the RTIG is to forward this message to the RTIAs of the other federates. The RTIA manages also queues for the other events of the simulation (`reflectAttributeValues()` and `receiveInteractions()`). When the delivery conditions are met, callbacks are executed with the `evokeCallback()` service, the last callback is TAG. In Fig.2, the separation and the link between the HLA services usage and their CERTI implementation though the exchange of messages is illustrated.

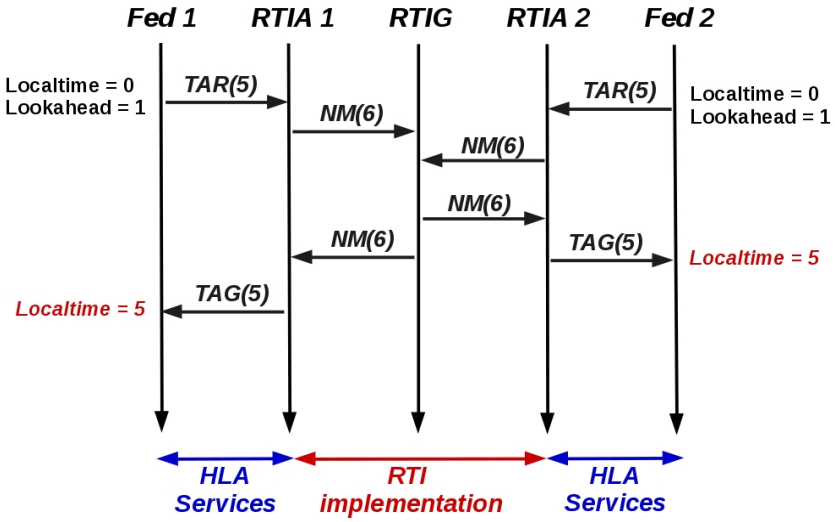

Figure 2. TAR/TAG message exchanges illustration for time-stepped federates

The following Fig.3 illustrates the flow of messages for event-driven fedetares and the fact that this first algorithm has some performances issues (and is not working with zero lookahead). The classical NULL message algorithm requires a number of messages which is a proportion between lookahead and the distance from current logical time. The messages are not all represented in the figure, it is possible to view the different scenarios discussed in the paper by using an interactive federate with traces. The interactive federate is one of our test cases available as an open-source software with CERTI.

[2]*equivalent to Greatest Available Logical Time (GALT)*

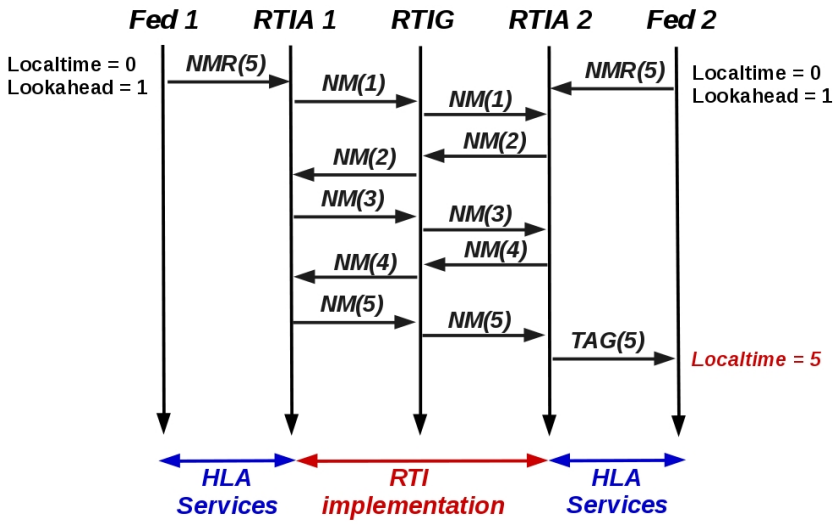

Figure 3. NMR/TAG message exchanges illustration for event-driven federates

2.4 CERTI 4.0 Evolutions

The new version of CERTI is, as usual, available at `https://savannah.nongnu.org/projects/certi`. The license is the General Public License (GPL) for the CERTI core and Lesser General Public License (LGPL) for the HLA interfaces. Following the needs of some users and our needs, we have added some missing services like the Management Object Model (MOM) and the modular Federation Object Model (FOM). Some services are still missing as an example for optimistic time management or for smart updates. The CERTI C++ code is now compliant with more recent version of the C++ standard (C++14) supporting per various compilers. This allows performance optimization but it has required some code review. We have added many tests and we propose three new test programs for all CERTI users (a modern billard, a MOM explorer and a new version of a Interactive Federate for the test and the understanding of the HLA services). Also, as the main purpose of this paper, this new version integrates optimization of time management services as described in the following chapter.

3. Optimization of Time Management in CERTI 4.0

3.1 The NULL Message Prime (NMP) protocol

A first version of this protocol was described in [14] and a modified version of this protocol is described in this paper. In particular, we discuss the relationship between the *time management* services and the *object management* services. The idea of our NULL Message Prime algorithm is to take advantage of the CERTI central component: the RTIG. In the classical NULL message algorithm the RTIG is only acting as a pure gateway which distributes the NULL message to each concerned federate. It does not even know the content of the message, nor the fact that Fed1 (resp. Fed2) is currently in a time advancing state. Now, if we make the RTIG aware of the federate state, i.e. whether it has called NER (*i.e.* the federate is *NERing*) or TAR (*i.e.* the federate is *TARing*) and we let the RTIG collect all requested times of the NERing federates then the RTIG can optimize time management.

When the federates are only involved in a time advance loop (*i.e.* all federates are *NERing*) then the algorithm is simple, when a federate is NERing it will send a NULL PRIME message to the RTI, which will compute an RTI-wide LBTS. Note that the RTI-wide LBTS computation includes the NULL and NULL PRIME message information, such that if some federate is TARing while other are NERing the protocol is still valid. Whenever the RTI-LBTS strictly increases, the RTI itself (in our case RTIG) will generate an anonymous NULL message Prime and broadcast it to all time constrained federates. When a federate (in fact its RTIA) receives an anonymous NULL message it will trigger the usual local LBTS computation. The message sequence chart corresponding to the previous case is given in Fig.4.
In this case the number of NULL messages exchanged before getting `TAG(5)` did go from 10 down to 4. However, the number of messages used by our NMP algorithm is independent of the NER value and the lookahead (including zero lookahead case) while classical NULL message algorithm requires a number which is a proportion between lookahead and the distance from current time. This description has been already done in our original paper but we haven't considered the case of the simulation message exchange. As stated before, we now have to talk about the link between the *time management* services and the *object*

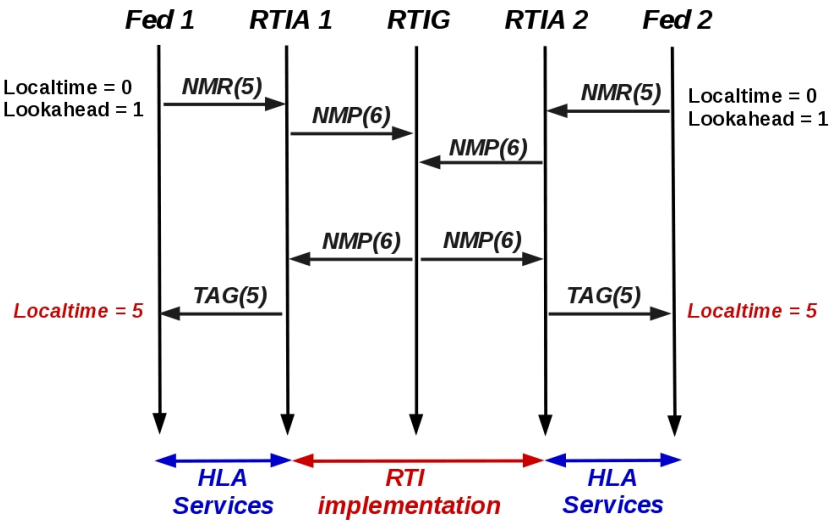

Figure 4. NMR/TAG message exchanges illustration for NMP protocol (event-driven federates)

management. Our original optimization was causing some deadlocks in some corner cases with federates exchanging time-stamped simulation messages by using `updateAttributesValues()` (UAV) or `sendInteraction()` (SI) services calls. As stated above, each federate has a contract with the RTI and it can't send a simulation message with a logical time-stamp smaller than its current logical time plus its lookahead. Therefore, it is allowed to send a message with any logical time stamp bigger than the lookahead. As illustrated in Fig.5, in the NMP algorithm, when a federate sends some simulation message, the time-stamp of Null Prime message will be the minimum value of the UAV/SI logical timestamps and the logical time-stamp in the NMR. Moreover, in order to handle complex cases (many TSO messages produced before NMR call), each RTIA process must store in a queue the timestamps of the TSO messages sent per the federate. This ensure a correct computation of the global minimum per the RTIG receiving successive NMP messages from the same RTIA during the time advance phase of the associated federate.

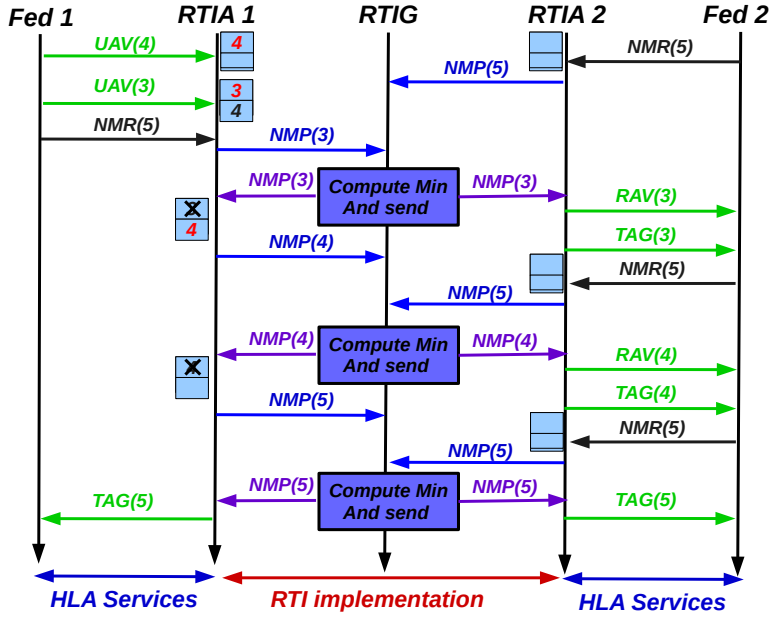

Figure 5. UAV/NMR/TAG message exchanges illustration for NMP protocol (event-driven federates)

The NMP algorithm co-exists with the classical NM algorithm; it only generates additional new NULL Message when enough information has been collected on the RTIA and the RTIG processes, where the RTIG sees every message exchanged inside the federation. A fully decentralized RTI may implement the same algorithm as soon as some broadcast protocol is available.

We think that the NULL Prime Message algorithm is somehow equivalent to global reduction based algorithm like the one from Mattern [21]. We think that our approach has several advantages (it needs to be confirmed per further investigations in future):

1) It is automatically triggered as soon as something is worth doing it. We do not have to look for the appropriate instant to start a wave/reduction.
2) We do not have to face the restart issue neither because even if transient message are in the network, the anonymous NULL message built by the algorithm is valid.
3) The number of message generated by the algorithm is constant and independent from lookahead value, including zero lookahead.

3.2 Different Techniques of Polling

Even if our applications have not exhibited a very important computing load for the RTIG process, its communication load is important. Its main processing loop is dedicated to receive a message from each connected RTIA process, to filter it according to the instructions of publish/subscribe and retransmit the messages to corresponding other RTIA processes. All these messages receptions and transmissions are done using network sockets I/O. Therefore the RTIG needs a polling method to use for the monitoring of all the events generated by all the connected sockets to RTIA processes. Historically, the RTIG was using the oldest solution with the well known `select()` service. Nowadays, Linux operating systems provide new methods called `poll()` and `epoll()` which can offer better performance under certain configurations and assumptions [22]. We implemented these two additional methods inside the RTIG and the user can use each of these methods (under Linux) by selecting the proper compilation flag. We are comparing these methods in the next section (see Table 1).

3.3 RTIG as a network server

As stated many times above, the RTIG is the central communication process of CERTI. However, on traditional network architecture, the RTIG is located on a computing node with a single network connection. The following figure (*Cf.* Fig. 6) is illustrating the deployment of a federation composed of 4 nodes running several federates (1 to N in each node) in a classical switched Ethernet local area network with a bandwidth of 1 Gbits/s (it can be 100 Mbits/s). We can see in this figure that the RTIG, running on node 5, has a single access to the network even if it is supposed to handle all communication flows coming from the whole federation (*i.e.* all the federates).

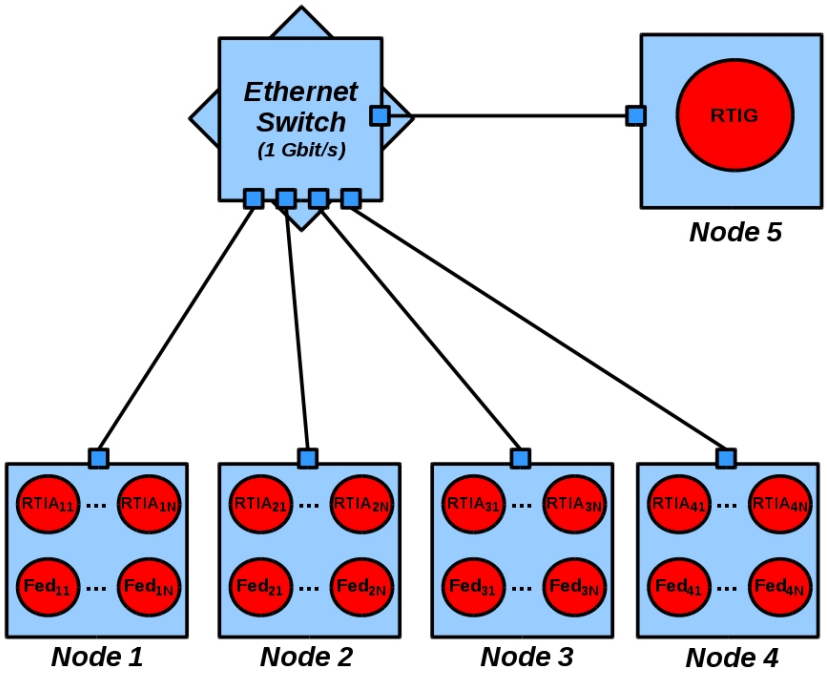

Figure 6. Deployment of CERTI federation over a classical Ethernet architecture

Therefore, in order to optimize CERTI performance, we thought to take advantage of intelligent software based switch. In other words, this specific device is a processor (mono/multi core(s)) with many Ethernet communication ports. As an example, the device we used for our tests has an Intel i5 cpu with 12 Ethernet ports (1 Gbits/s capables). Using such device, we can build

architectures such as the one depicted in Fig. 7 where the RTIG process is used a central intelligent software based Ethernet switch. We are experimenting the use of this new architecture in the next section (see Tables 2 and 3).

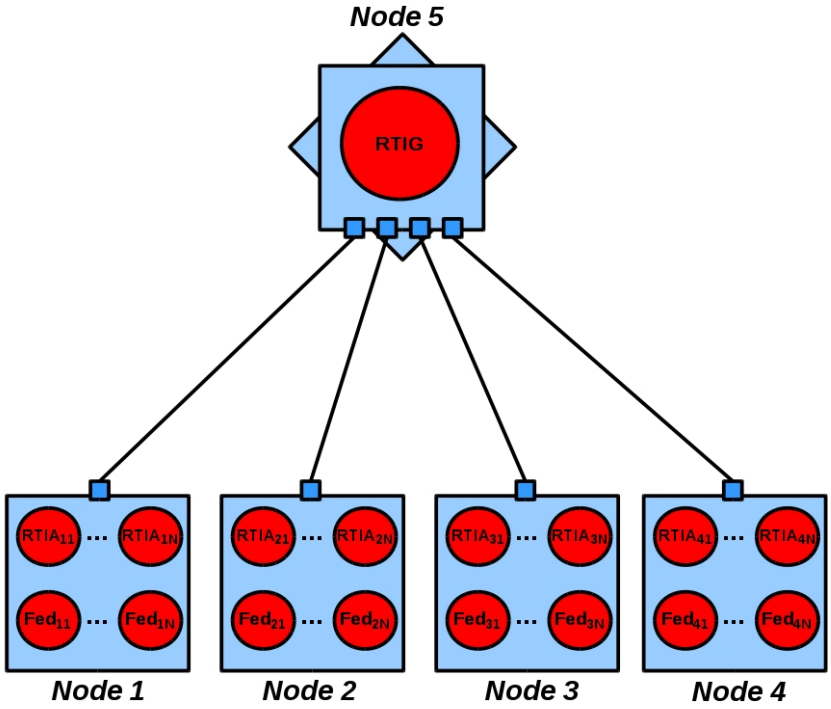

Figure 7. Deployment of CERTI federation over the new Ethernet architecture

4. Time Management Analysis

4.1 A new open-source benchmark

Time management performance analysis will be done from a new benchmark program. Some unit testing programs have been proposed by DMSO back in the days but are no longer available or maintained (to our knowledge). For HP-CERTI [12], we used the billiard program CERTI that can go as fast as HLA allows but all billard federates are time-stepped federates (using TAR). Therefore, for next-event federated tests and especially NMP, we had to develop a new test program. This open-source benchmark is available here `https://sourceforge.isae.fr/projects/benchmark_hla`. The idea behind is to allow the user to multiply the number of federates. The user can call a federate playing with multiple parameters:

- The *DataSizeByte* is the size in bytes of the data sent per the federate in an UAV call;
- The *lookahead* is the contract for conservative time advance as explained before;
- The *timestep* is the maximum for the time stamp of federate next update;
- The *starttime* is the starting logical time of federate;
- The *endtime* is the overall ending logical time for the simulation;

Considering that t_{fed} is the current federate localtime, the benchmark federate main loop is build around the following actions:

1) Choose a random time t_{uav} between $t_{fed} + lookahead$ and $t_{fed} + timestep$, mathematically:
$$t_{uav} \in]t_{fed} + lookahead; t_{fed} + timestep]$$
2) Produce an `updateAttributeValues()` event with this timestamp: UAV(t_{uav});
3) Ask to advance in logical time by `nextMessageRequest()` to the end of simulation logical time: NMR($endtime$);
4) wait for the `timeAdvanceGrant()` (TAG agreement) with `evokeCallbacks` service.

All federates in the benchmark are both regulators and constrained federates for the time advance services. To ensure a proper initialization of the whole federation (starting all the federates), we use synchronization point services to get an initial trigger and launch all the federates and the "same" time. The creator of the federation (the one which got a successful `createFederationExecution()`) will register the total execution time on its local cpu clock (which is a reference to the wall clock time). We can compare then the actual times of execution for the federation to go until the logical $endtime$. This generic implementation allow to easily execute and compare different configurations.

4.2 Results

We include in this part some of our first experimental results obtained by running our benchmark under different configurations. Table 1 shows the results obtained between original NM algorithm (using `select()`) and our NMP algorithm using the different implemented polling methods (`select()`, `poll()` and `epoll()`). The benchmark has been executed on a single computer equipped with an i7-6700 (octo-core processor @3.40GHz with 16 GB RAM memory). The configuration of the benchmark (for each federate) is the following:

- $DataSizeByte = 1000$ (bytes)
- $lookahead = 0.01$
- $timestep = 1.0$
- $starttime = 0.0$
- $endtime = 1000.0$

Methods	4 federates (seconds)	8 federates (seconds)	12 federates (seconds)	16 federates (seconds)	20 federates (seconds)	40 federates (seconds)
Null Message (select)	$7,95$	$27,69$	$57,71$	$100,28$	$158,12$	$776,89$
Null Message Prime (select)	$2,59$	$7,37$	$13,86$	$22,28$	$32,28$	$156,52$
Null Message Prime (poll)	$2,48$	$7,23$	$13,58$	$21,24$	$31,65$	$154,34$
Null Message Prime (epoll)	$2,50$	$7,29$	$13,74$	$21,92$	$31,81$	$155,52$

Table 1. Measurements of the different methods on a single host

Table 2 and table 3 show the results obtained under the two distributed architectures described before: (1) the standard Ethernet switch (see Fig.6) and (2) the software based central RTIG architecture (see Fig.7). The benchmark parameter assignment is identical to the one described before.

Methods	4 federates (seconds)	8 federates (seconds)	12 federates (seconds)	16 federates (seconds)	20 federates (seconds)	40 federates (seconds)
Null Message (select)	$29,78$	$50,34$	$69,87$	$113,61$	$165,37$	$812,52$
Null Message Prime (select)	$9,36$	$16,51$	$27,93$	$42,90$	$60,52$	$237,13$
Null Message Prime (poll)	$8,87$	$15,91$	$28,13$	$39,93$	$60,43$	$238,77$
Null Message Prime (epoll)	$9,13$	$16,73$	$30,95$	$43,71$	$62,45$	$239,32$

Table 2. Measurements of the different methods on a classical Ethernet architecture

Methods	4 federates (seconds)	8 federates (seconds)	12 federates (seconds)	16 federates (seconds)	20 federates (seconds)	40 federates (seconds)
Null Message (select)	$26,42$	$40,39$	$64,13$	$104,71$	$143,13$	$789,27$
Null Message Prime (select)	$8,75$	$13,41$	$17,68$	$29,95$	$38,47$	$143,35$
Null Message Prime (poll)	$8,51$	$12,57$	$16,60$	$28,72$	$36,28$	$140,73$
Null Message Prime (epoll)	$8,21$	$12,82$	$18,41$	$29,28$	$37,36$	$142,15$

Table 3. Measurements of the different methods on the new Ethernet architecture

These first experimental results illustrate the benefits of our Null Message Prime (NMP) algorithm over different execution infrastructures. We noticed that the execution of the benchmark on a single host is more efficient than on distributed architectures and we claim that this is due to the non-existing processing need of a benchmark federate (it does not embed any simulation algorithm logic as the goal of the benchmark is to test the efficiency of the HLA time management services). During our

experiments, we have also observed that the `poll()` method was the more efficient (in average); however, the three polling methods offer very similar performances for CERTI. Furthermore, the new proposed distributed architecture for CERTI looks promising and we need to confirm this first impression by investigating further using different hardware technologies for the software based Ethernet switch (running the central RTIG process).

5. Conclusion and Perspectives

In this paper, we have presented a detailed analysis of HLA time management services implemented in CERTI. We have highlighted some possible improvements for these services in CERTI (especially the NMP algorithm) and implemented these solutions. We have experimented the proposed approaches and the results obtained and presented in this paper can be reproduced because we have released the whole software package as open-source. Current CERTI 4.0 performances are very good for our real-time and/or high performance simulations applications. We are still continuing our effort to increase the high-performance and real-time properties of CERTI and thereby to ensure better responsiveness of all HLA services it offers. In this sense, we are currently working on a new version of the original CERTI HLA inter-federations bridge [11] which will also be released as an open-source software this fall. We plan to use this new bridge to build more complex architecture with multiples RTIG processes running as software based Ethernet routers. All the results presented are reproducible because all software parts we used in this article are release as open-source (GPL, LGPL). Indeed, the current CERTI ecosystem is very complete and contain a lot of test cases and applications. In addition, we have also released an open-source of our flight simulator called *OpenSDSE* [17] available here `https://sourceforge.isae.fr/projects/opensdse`. To resume, CERTI 4.0 is built on a solid experience with HLA, supports a lot of existing applications and is ready for any new user.

References

[1] 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.

[2] 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.

[3] 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.

[4] R.M. Fujimoto: “Time Management in the High Level Architecture”, Simulation, Vol.71, pp 388-400. December 1998.

[5] D. Jefferson: “Virtual Time”, International Conference on Parallel Processing ICPP’83, Columbus, Ohio, USA, August 1983.

[6] P. Siron: “Design and Implementation of a HLA RTI Prototype at ONERA”, 1998 FallSimulation Interoperability Workshop, Orlando, September 1998.

[7] B. Bréholée, P. Siron: “CERTI, Evolutions of the ONERA RTI Prototype, 2002 Fall Simulation Interoperability Workshop, Orlando, September 2002.

[8] E. Noulard, J.-Y. Rousselot, P. Siron: “CERTI: an open Source RTI, why and how, 2009 Spring Simulation Interoperability Workshop, San Diego, March 2009.

[9] P. Bieber, J. Cazin, P. Siron, G. Zanon: “Security Extensions to ONERA HLA RTI Prototype, 1998 Fall Simulation Interoperability Workshop, Orlando, September 1998.

[10] M. Adelantado, P. Siron: “Multiresolution Modeling and Simulation of an Air-Ground Combat Application, 2001 Spring Simulation Interoperability Workshop, Orlando, March 2001.

[11] B. Bréholée, P. Siron: “Design and Implementation of a HLA Inter-federation Bridge, European Simulation Interoperability Workshop, Stockholm (Sweden), June 2003.

[12] M. Adelantado, J.-L. Bussenot, J.-Y. Rousselot, P. Siron, M. Betoule: “HP-CERTI: Towards a high Performance, high Availability Open Source RTI for Composable Simulations, Fall Simulation Interoperability Workshop, Orlando, September 2004.

[13] J.-B. Chaudron, E. Noulard, P. Siron: “HLA High-Performance and Real-Time Simulation Studies with CERTI”, Proceedings of the European Simulation and Modelling Conference, October 2011.

[14] 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, Boston, United States, April 2011.

[15] J.-B. Chaudron, D. Saussié, P. Siron, M. Adelantado: “How to solve ODEs in real-time HLA distributed simulation.”, Proceedings of the Fall Simulation Interoperability Workshop, ORLANDO, United States, September 2016.

[16] H. Deschamps, G. Cappello, J. Cardoso, P. Siron: “Toward a formalism to study the scheduling of cyber-physical systems simulations”, Proceedings of the IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications, Roma, Italy, October 2017.

[17] J.-B. Chaudron, D. Saussié, P. Siron, M. Adelantado: “Real-time distributed simulations in an HLA framework: Application to aircraft simulation”, Simulation, Vol.90, Issue 6, pp 627-643, June 2014.

[18] G. Lasnier, J. Cardoso, P. Siron, C. Pagetti, P. Derler: “Distributed Simulation of Heterogeneous and Real-time Systems, 17th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Delft (Netherlands), October 2013.

[19] C. Michel, J. Cardoso, P. Siron: “Time Management of Heterogeneous Distributed Simulation, Proceedings of the 31st European Simulation and Modelling Conference (ESM’2017), Lisbon, Portugal, October 2017.

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

[21] F. Mattern: “Efficient algorithms for distributed snapshots and Global Virtual Time approximation”, Journal of Parallel and Distributed Computing, 1993.

[22] L. Gammo , T. Brecht , A. Shukla , D. Pariag: “Comparing and evaluating epoll, select, and poll event mechanisms”, In Proceedings of 6th Annual Linux Symposium, Ottawa, Canada, 2004.

Author Biographies

JEAN-BAPTISTE CHAUDRON received his PhD in 2012 from ISAE-SUPAERO. He works at ISAE-SUPAERO for the Department of Complex Systems Engineering (DISC) as a research engineer. His fields of interest include parallel and distributed simulation, time-triggered and real-time systems and scheduling theory.

PIERRE SIRON graduated from a French High School for Engineers in Computer Science (ENSEEIHT) in 1980, and received his doctorate in 1984. He works in parallel and distributed systems and he is leader of the CERTI project. He is Professor at the ISAE-SUPAERO and the head of the computer science program of the ISAE-SUPAERO formation.

MARTIN ADELANTADO graduated from a French High School for Engineers in Computer Science (ENSEEIHT) in 1979, and received his doctorate in 1981. He is an ONERA (French Aeronautics and Space Research Center) Research Engineer and works at the Information Processing and Systems Department (DTIS). His fields of interest include simulation, real-time systems and distributed systems.

Acknowledgment

The authors would like to thank **Clément Vannier** from SCALIAN for his great work on parts of the CERTI 4.0 extensions.