Design and modeling techniques for real-time RTI time management

Jean-Baptiste Chaudron
Eric Noulard
Pierre Siron
ONERA/DTIM
2 avenue E. Belin
31055 Toulouse Cedex
France
jean-baptiste.chaudron@onera.fr, eric.noulard@onera.fr, pierre.siron@onera.fr

Jean-Baptiste Chaudron
Pierre Siron
Université de Toulouse, ISAE
10 avenue E. Belin
31055 Toulouse Cedex
France
pierre.siron@isae.fr

Keywords:
High Level Architecture, Distributed Simulation, Modeling, Time Management algorithms

ABSTRACT: *The design and implementation of an RTI includes several HLA services, a very important one being the time management. Several time management protocols exist ranging from the Chandy-Misra-Bryant (CMB) null message conservative algorithm up to optimistic Jefferson time warp one. We are interested in enhancing the high- and/or real-time performance of our open source RTI (CERTI) including the time management protocol. In order to achieve these goals we explored two complementary approaches, first, design a new conservative time management algorithm which avoid the time creep problem of classical CMB and then apply model-checking techniques and tools to specific real-time federation in order to formally ensure the validity and the complexity of the algorithms on those federations. We will present in this paper a new conservative time management algorithm and the beginnings of model-checking techniques applied to predict and ensure the real-time performance of specific federation.*

1. Introduction

The goal of any simulation is to perform calculations on a model (or set of models) representing a concrete system of the real world. The real system that we want to study using simulation always respects two key principles:
(1) The *determinism principle* : the future of the system can be determined from its present state and its past. In other words, at any time $\boldsymbol{t}$, there is an $\boldsymbol{\varepsilon}$ value for which the future behavior of the system at $\boldsymbol{t + \varepsilon}$ is exactly known.
(2) The *causality principle* : the future never influences the past. Specifically, the system state at time $\boldsymbol{t}$ is independent of anything that may occur at a time $\boldsymbol{t'}$ greater than $\boldsymbol{t}$.

These two principles govern the evolution of real systems such that, any simulation of a such a system must respect the determinism principle and the causality principle. This last point and the technics used to respect it are the key interest of this article. Furthermore, in event-driven simulation (like an HLA one), every simulator must determine the next instant, in the simulated time, which will produce a state change in the whole system. At that point, it must perform the treatment describing the state change and amends the simulated time. In the distributed context, the simulators are located on different processors so that they cannot know the overall system state. They must make their decisions concerning the progression in the simulated time from local state, possibly calculated using the contents of messages it has received from other simulators.

These mechanisms used to advance simulated time are specified by the HLA standard **[1][2][3]** through time management services. Those services help to ensure a coherent global behavior (in terms of simulated time) by using different types of technics and algorithms provided by the HLA middleware: the Run Time Infrastructure (RTI).

Middleware in computing terms is used to describe a software agent acting as an intermediary layer between different distributed processes. This software has to be seen in the domain of interoperability. It is a connectivity software which enables the execution of several interacting applications on one or more networking computers. The RTI (Run-Time infrastructure) is a middleware that is required when using HLA compliant simulation (see Figure 1). The RTI is the fundamental component of HLA. It provides a set of software services that are necessary to support federates coordinated operations and data exchange during an execution. In other word, it is the implementation of the HLA interface specification **[1][2][3]** but is not itself part of the specification.

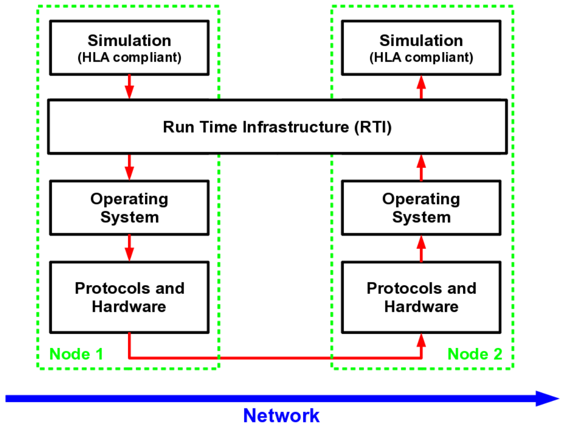

Figure 1: RTI Middleware Illustration

In our experiment, we choose to use CERTI Open Source RTI **[4]** developed by its open source community **[5]** maintained by ONERA. It is a RTI which 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. The CERTI architecture is depicted in Figure 2. Each federate process interacts locally with an 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/broadcast of messages to all RTIA. We will see in §4 how we can take advantage of this central entity for developing a new time conservative algorithm.

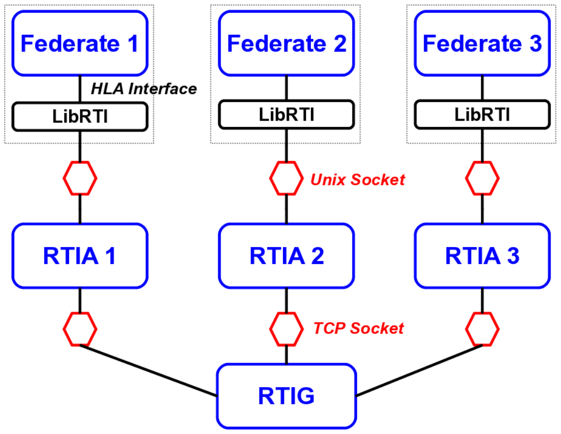

Figure 2: CERTI architecture

A key benefit of mastering the implementation of the RTI is to be able to incorporate changes in its source code, thus one may tailor it for specific research needs. In this work, we will explain the use of CMB algorithm, also called *NULL Message Algorithm*, in CERTI Run Time Infrastructure for real-time and high performance simulations.

The outline of this paper is the following: Paragraph 2 describe the different time management services and algorithms used in distributed simulation research community. In paragraph 3, we study the use of a NULL Message algorithm in the case of real-time periodic federates and present analytical methods to quantify the NULL messages exchange during the simulation run time. In paragraph 4, we will present a new algorithm, named "*NULL Message Prime algorithm*", based on the original one of Chandy and Misra which leverages the particular CERTI architecture. After that, we present model-checking models to formally verify the behavior and the validity of our algorithms. Finally, we give some concluding remarks and expected perspectives for future works.

2. Time Management background

2.1 General view

Time management mechanisms, inherited from ALSP standard **[6]** and provided by the HLA middleware, are one of the main benefits of this simulation standard **[7]**. These services allow a consistent global time throughout the whole simulation by using different methods and algorithms. Specifically, each simulation message is assigned a time-stamp, and the RTI ensures that messages are delivered to each federate in time-stamp order, and no message is delivered to a federate in its past. The main operation required to implement time management services is the determination of the *GALT Greatest Available Logical Time (*also called

LBTS *Lower Bound on Time-Stamp* for HLA 1.3 standard **[1]**), of each federate. The GALT value is crucial because any message with time-stamp less than GALT can be delivered to the federate while still guaranteeing time-stamp order delivery. The HLA standard does not provide or advise a specific implementation for this Time Management service offered by the RTI. Two types of approaches, which ensure the causality constraint, have been proposed in the literature :

- **The optimistic strategy (or coherent-post)**: each message is processed by the simulator in order of their arrival until it detects a violation of the local causality constraint. In this case, the simulation requires a mechanism for turning back (*roll-back* mechanism). It is a strategy based on the concept of virtual time initially described by Jefferson **[8]**.

- **The conservative strategy**: which avoid the violation of the local causality constraint altogether. Two generations of conservative strategy and corresponding algorithms are usually used in RTI implementations as explained in § **2.2**.

2.2 Conservatives strategies in HLA and limitations for real time and high performance simulations

The first generation time management services are based on the so-called "NULL Message Algorithm" (NMA) of Chandy and Misra **[9].** This is the main algorithm implemented in CERTI and it is used to avoid deadlock in a conservative federation like the one illustrated in Figure 3 (extracted from Richard Fujimoto works).

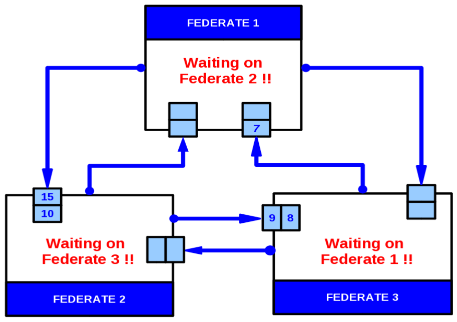

Figure 3: Deadlock example

This approach is based on a contract for each federate called *lookahead*. Each federate undertakes not to send simulation messages with a time stamp less than its local time plus its *lookahead*. The respect of this contract enables the exchange of additional messages called *NULL messages* (messages containing only time-stamps) indicating the Lower Bound on the Time Stamp (LBTS equivalent to GALT) of future messages it could send (see Figure 4).

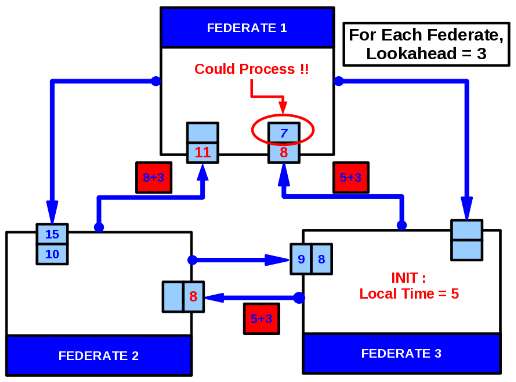

Figure 4: Classical Null Message example to avoid Deadlock

The main shortcoming of this approach for real-time or/and high performance simulation is the communication overhead implied by additional exchanges of NULL messages between all simulators. Furthermore, if the lookahead parameter is not well chosen, the simulation is subject to ***lookahead time creep*** problem (see §2.3): the number of NULL messages may become unacceptable and limits the performance of the simulation.

The second generation time management services is based on different kind of algorithms. Those algorithms are not implemented in current CERTI version but the new algorithm proposed in §4 is somehow equivalent. The second generation algorithm are not subject to the time creep problem. The GALT is calculated, at the request of each federate, by establishing a "picture" (snapshot) of the overall state of the distributed simulation. Yet, these kind of algorithms are subject to another key issue called "*transient message problem*" which mean there exist any message which has been sent but not yet received. Samadi's algorithm **[10]** consider these message using acknowledgment messages to solve it. Mattern's algorithm **[11]** use a coloring scheme and a vector counter method to account these transient messages. However, for real time purpose, computation of GALT value realized by these algorithms cannot generally be guaranteed to complete within a bounded time. The reason comes from the fact that it depends on the participation of all other federates in the execution while transient messages can cause a GALT computation to be aborted and retried. Fujimoto and McLean have modified the LBTS computation in order to respect a bounded time computation for real-time executions **[12]**. The authors have then proposed an extension for time-stamp assignment in Time Management mechanisms called "*the offset-epoch* *method*" to increase the efficiency of the original time management algorithm by eliminating these transient messages and computing an LBTS adapted to this new method **[13]**.

2.3 The Lookahead Time Creep problem

As we said previously, the performance of a TM algorithm depends on the lookahead of the simulation application **[14]**. With a small lookahead, a prominent problem of asynchronous TM algorithms occurs, which is known as "lookahead time creep" where some (if not all) federate can only advance their logical time by steps of a lookahead increment before the next non-NULL message in the simulation application can be processed (see Figure 5).

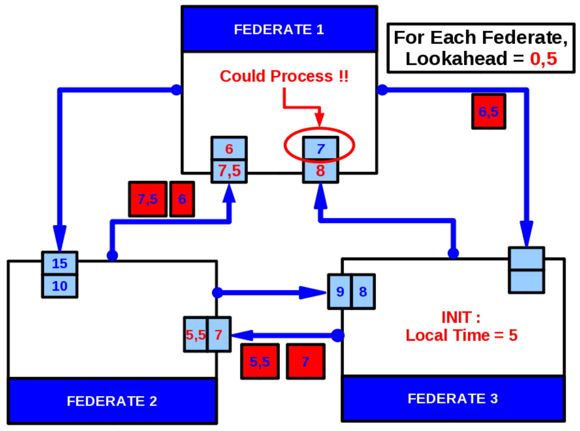

Figure 5: Time Creep Example with Inappropriate lookahead

ONERA studies show that time management using Chandy Misra Bryant seems good for a particular type of real-time federates (see **[17]** and §3.1).

2.4 Different types of Time Management services

To respect causality principle, RTI guarantees that all federates will not receive any simulation messages with a time stamp less than its logical time. Various services exist to allow the federate to express its requests for advancing its local logical time:

- `TimeAdvanceRequest`(TAR),
- `TimeAdvanceRequestAvailable`(TARA),
- `NextEventRequest`(NER),
- `NextEventRequestAvailable`(NERA),
- `FlushQueueRequest` (FQR).

TAR and TARA are devoted to federates that internally employ a time-stepped mechanism which are the type of interest in §3. NER and NERA are suitable for federates who run with event-driven mechanism (analyzed in §4). The third type of service is an additional service devoted to time advance with optimistic strategy which is not the main interest of this work. CERTI implements TAR, TARA, NER, NERA using NULL message algorithm (including the zero-lookahead case **[15]**) and further optimize NER and NERA case using NULL PRIME message algorithm presented in §4.

3. CERTI Time Management for real time periodic federates

3.1 Periodic federates vision

The concept of periodic federates, named "*repeatability within simulation*s" has been introduced by Fujimoto and McLean **[15] [16]** with their works on real-time and distributed simulations. Federates, involve 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 : (1) a reception phase, (2) a computation phase, (3) a transmission phase and (4) a slack time phase. ONERA and CNES studies **[17]** show the necessity of adding a synchronization phase (see Figure 6) to ensure the global coherent run time of the whole simulation.

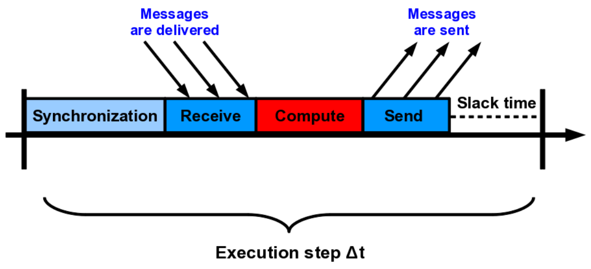

Figure 6: illustration of periodic federate cycle

Historically, in DIS simulation standard **[18]**, 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 whole global simulation run-time. In Fujimoto and McLean works, synchronization and reception phases are made in the same time by time management mechanisms (see § 4.2). 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) **[19]** system for distributed applications available on our Linux Red Hawk platforms **[20],**

2. The federate which have the high speed cycle sends an interaction to all each 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. The time advance can be correlated to an hardware clock to ensure the respect of real time constraints.

This synchronization phase is essential in the distributed context where the different nodes do not have a shared time reference (each using its own local wall-clock time). For present work, we simply consider periodic federates synchronized by time management mechanisms.

ONERA and CNES experiments **[17]** with federates based on the same periodic pattern, show that first generation time management services (see §3.1) implemented in CERTI, seems good for real-time. Indeed, best results are obtained by requiring time management services in both tests cases of study. In fact, the overhead is compensated by the better synchronization that these services enforce between federates. This better synchronization between federates reduces latency in data exchanges, reduces the cycle duration and makes the global behavior more regular because of great jitters reduction. In fact, the NULL message algorithm implemented in CERTI enforces a very good synchronization.

3.2 Reception phase with time management services

During each Δt step, a real-time periodic federate asks their logical time advance to RTI and also all the simulation messages available by using the `timeAdvanceRequest()` service (noted `TAR()`). The RTI allows this logical time advance (with respect to causality constraint) by sending all the `reflectAttributeValue()` callbacks (noted `RAV()`) available for the logical time asked and finishes by accept the time advance by using the `timeAdvanceGrant()` callback (noted `TAR()`). These different mechanisms, form the synchronization and reception phase of a periodic federate using time management mechanisms (see Figure 7).

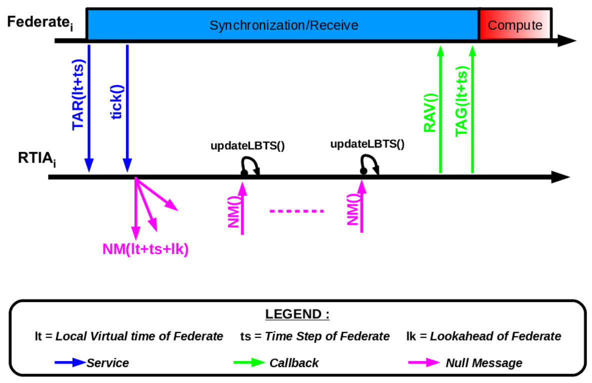

Figure 7: illustration time management services used by a periodic federate

3.3 Quantify Messages Exchange on Whole simulation

For real time simulation purpose, we want to ensure predictability of any service used. In order to add determinism to first generation time management technics involved in CERTI software, we propose an analytical methodology to formally quantify the number of null messages exchanged between each real time periodic federates involved in a real time simulation.

However, to apply our method, we need to make some basic assumptions:

1) The global simulation (Federation in HLA terms) is composed by ***N*** real-time periodic federates indexed by an integer ***i €[1, ... , N]*** ;
2) We consider now a federate noted ***Fed(i) :***
   1. its logical time is noted ***t(i)***;
   2. its time step (the expression of its computational periodicity in the simulation time) is noted ***ts(i);***
   3. its communication step (the expression of its communication periodicity in the simulation time) is noted c***s(i);***
   4. its look ahead is noted ***lk(i)***;
3) The $PS(i,j)$ is a matrix of size NxN to identify the different communications between the various simulators (equivalent to publisher-subscriber informations).
4) The TS_{LCM} value is the *least common multiple (LCM)* of of all federate's time step;
5) The interval study, expressed in simulated time, is usually equal to TS_{LCM} .

The number of NULL message $NM_S(i)$ sent by a federate ***Fed(i)*** beginning with ***t(i)=0*** and concluding by ***t(i)*** = TS_{LCM} is :

$$\boxed{NM_S(i)=\frac{TS_{LCM}}{ts(i)}} \quad (1)$$

with i ***€[1, ... , N] .***

Reciprocally, the number of NULL message $NM_R(i)$ received by a federate ***Fed(i)*** beginning with ***t(i) =0*** and concluding by ***t(i)*** = TS_{LCM} is :

$$\boxed{NM_R(i)=\sum_j\left(\frac{TS_{LCM}}{ts(j)}\right)} \quad (2)$$

with i,***j €[1, ... , N]*** and ***i≠j .***

The number of simulation messages $SimM_S(i)$ sent by a federate ***Fed(i) beginning*** with ***t(i) =0*** and concluding by ***t(i)*** = TS_{LCM} is :

$$\boxed{SimM_S(i)=\frac{TS_{LCM}}{cs(i)}} \quad (3)$$

with i ***€[1, ... , N] .***

Reciprocally, the number of NULL message $SimM_R(i)$ received by a federate ***Fed(i)*** beginning with ***t(i) =0*** and concluding by ***t(i)*** = TS_{LCM} is :

$$\boxed{SimM_R(i)=\sum_j\left(PS(i,j)\times\frac{TS_{LCM}}{ts(j)}\right)} \quad (4)$$

with ***i,j €[1, ... , N]*** and ***i≠j .***

3.3 Quantify NULL Message received between TAR and TAG

We now present a analytical method that could be called by one real time periodic federate to evaluate the number of NULL Message received (see Figure 8) between its ask of time advance using `TAR` service and its validation by the CERTI RTI using `TAG` callback (explicitly made by RTIA process which realizes LBTS computations).

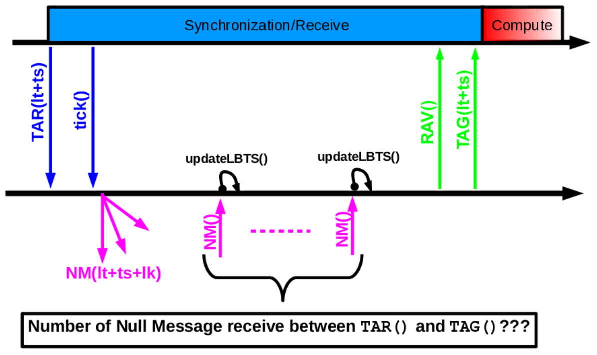

Figure 8: Number of Null Messages between `TAR()` and `TAG()`

In order to apply this method for any federate involved in the whole federation, we need to make some additional assumptions:

1. Each federate (noted ***Fed(i)***) have to locally maintain a static vector noted ***TimeStepFederation(i)=(ts(1), ...,ts(N))***. This allows the federate to know others federates cycles.
2. Each federate (noted ***Fed(i)***) have also to maintain a dynamic vector noted ***GlobalStateFederation(i)=(gt(1), ..., gt(N))***. Each gt is equal to local time plus time step plus lookahead of each federate (***gt(i) = lt(i) +ts(i)+lk(i)***). This vector is currently updated during the simulation by NULL Message exchange and LBTS value is computed from this vector.

We consider a federate ***Fed(i).*** Under these basics assumptions, it could compute the number of NULL message (noted $NM_{Cycle}(i)$) it have to receive between its `TAR` and its `TAG`.

$$\boxed{NM_{Cycle}(i)=\sum_j W_j} \quad (5)$$

and

$$\boxed{W_j=\left\lceil\frac{t(i)+ts(i)-gt(j)}{ts(j)}\right\rceil} \quad (6)$$

for ***i,j €[1, ... , N]*** and ***i≠j .***

Note that $\lceil x\rceil$ denote the largest integer taller or equal to x

3.4 Asynchronous problematic

The traditional Chandy-Misra-Bryant NULL Message algorithm is a typical distributed asynchronous algorithm. The different formulas describe in § 3.3 have to be adapted for taking into account this problematic.

For example, when a federate ***Fed(i)*** compute the number of null message it have to receive between its `TAR` and its `TAG` (noted $NM_{Cycle}(i)$). Others federates could ask their time advance for their next cycles before the RTI allows the time advance for ***Fed(i).*** The NULL message algorithm apply to real-time periodic federates ensure that all federates could not have more than one cycle difference between each others. So, precedent formula (5) have to be adapted for asynchronous world. In this case, we could ensure a minimal and a maximal born for $NM_{Cycle}(i)$ computations :

$$\boxed{\sum_j W_j \leq NM_{Cycle}(i) \leq \sum_j W_j + (N-1)} \quad (7)$$

W_k computed by formula (6) and ***i,j €[1, ... , N]*** and ***i≠j .***

4. Time Management for real-time event driven federate

In this section, we are concerned with federation which contains at least two federates which use NER or NERA time advancing method. Note that beginning with IEEE-1516-2000 revision **[2]** of the HLA standard NER and NERA services are now called NMR (`NextMessageRequest`) and NMRA (`NextMessageRequestAvailable`). Besides the different names the services are identical. We must recall that for effectively advancing time the HLA standard indicates that the federate shall call one of the time advance service **and** then give time to the RTI to process the request before receiving the `TimeAdvanceGrant`(TAG) callback. Before IEEE-1516-2010 **[3]** the federate has to call `tick/evoke` service in order to be able to receive the TAG callback, this is called the EVOKE model. CERTI does not implement the alternative IMMEDIATE model so we will not treat this case.

4.1 NULL Message algorithm CERTI implementation

The implementation of the CMB NULL message algorithm in CERTI has been mapped to the CERTI architecture (see figure 2). The CERTI LRC called RTIA contains the heart of the time management algorithm. When the federate call the NERx service the RTIA is responsible for sending NULL message to all time constrained federates in the federation. In fact, when the federate invokes the RTI using `tick/evoke` its RTIA will send a NULL message to the RTIG process which will broadcast the message to all other time constrained federates. Each time an RTIA receives a NULL message from another federate he will recompute its local LBTS.

4.2 NER, NERA and time creep problem

The lookahead time creep problem is shown one more time as a message sequence chart presented in figure 9. Two federates "Fed1" and "Fed2" with lookahead=1 call the NER(5) service. They are alone in the federation so that they could theoretically advance their local time strait to instant t=5 but the classical NULL message algorithm gives the following sequence.

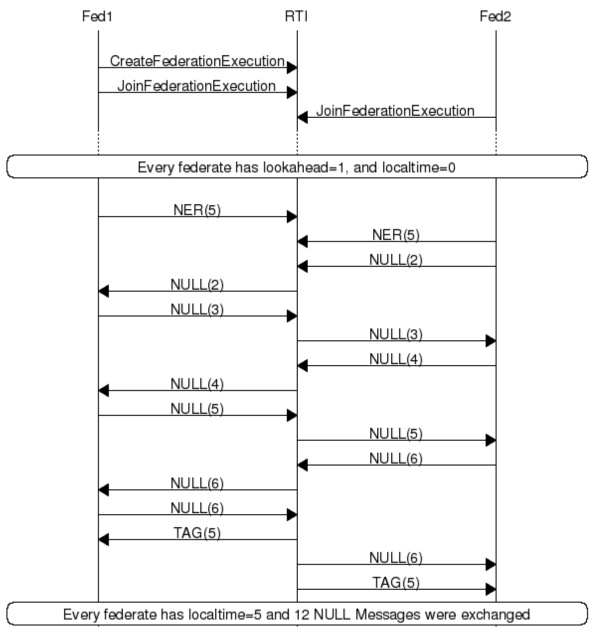

Figure 9: Time Creep problem

The time creep problem comes from the fact that each federate cannot send NULL message whose time stamp is bigger that its current local LBTS plus its lookahead. The local LBTS of each federate is growing at the pace of their lookahead. The smaller the lookahead the greater the number of NULL messages will be.

4.3 Use Central CERTI architecture

As it can been seen on the figure 9 we abstracted away the RTIA/RTIG distinction because the federates are not aware of this architectural choice. In the same way, the message sent to the RTI consist in reality of a call to the libRTI which sends a message to RTIA which in turns sends to RTIG. The full request/answer path is shown in figure 10.

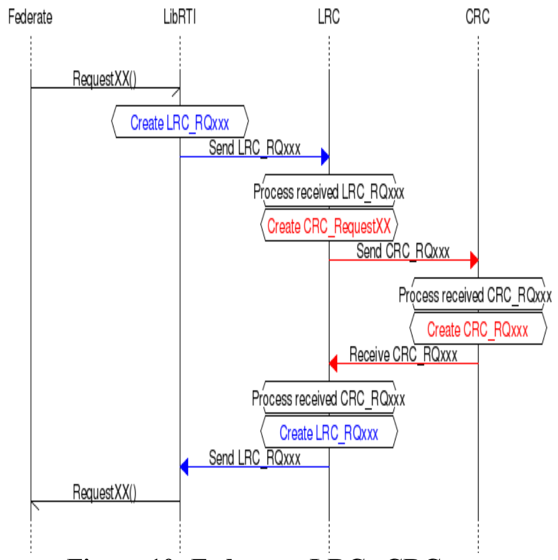

Figure 10: Federate--LRC--CRC

The idea of our NULL PRIME Message algorithm is to take advantage of the CERTI CRC, 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. He does not even know the content of the message, nor the fact that Fed1 (resp. Fed2) is currently in a time advancing loop. Now, if we make the RTIG aware of the status of the federate, i.e. whether he has called NER (he is NERing) or TAR (he is TARing) and we make it collect all requested date of the NERing federates then may be RTIG can do a better job.

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 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 figure 11.

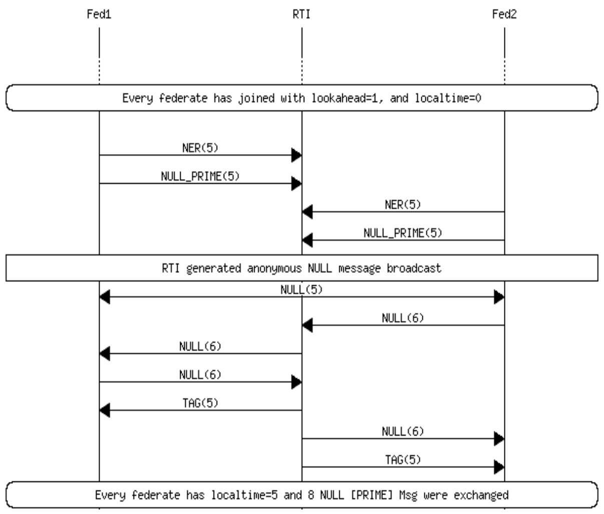
Fed1
RTI
Fed2
Every federate has joined with lookahead=1, and localtime=0
NER(5)
NULL_PRIME(5)
NER(5)
NULL_PRIME(5)
RTI generated anonymous NULL message broadcast
NULL(5)
NULL(6)
NULL(6)
NULL(6)
TAG(5)
NULL(6)
TAG(5)
Every federate has localtime=5 and 8 NULL [PRIME] Msg were exchanged

Figure 11: NULL PRIME Message

In this case the number of NULL [PRIME] message exchanged before getting TAG(5) did go from 12 down to 8. However, the number of message used by NULL PRIME 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.

The NULL PRIME Message algorithm co-exists with the classical NULL Message, it only generates new anonymous NULL Message when enough information has been collected. The new algorithm was relatively easy to implement in CERTI because we have the RTIG which “sees” every message exchanged inside the federation. 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 **[11]**. Our algorithm has several advantages:

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.

5. Conclusion and Perspectives

We have presented two important improvements concerning real-time HLA simulation. The first for multi-periodic time stepped federation for which we exhibit a formula which bounds the number of messages exchanged, thus ensuring performance. The second concerns event-driven federation for which we propose a new conservative time management algorithm, the NULL PRIME message algorithm which exhibits very interesting properties, including a solution to the time creep problem.

The next step of our work will be to finish the UPPAAL modeling in order to exhibit formal proofs. Real systems modeled by HLA simulations may have a discrete modeling. These systems are characterized by a given state, and its behavior over time can be described by a sequence of state transition. Therefore, we were interested in formalism of Finite Automata and Temporized Automata **[21]** and we have worked with the UPPAAL tool **[22]** to validate our approach for each part of the problem. The current UPPAAL models for formally validating the result of paragraphs 3 and 4 are an on-going effort

6. References

[1] Defense Modeling and Simulation Office DMSO: ***"High Level Architecture Interface Specification"***, Version 1.3 NG, Washington D.C., 1998.

[2] 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, 2000.

[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] B. Bréholée, P. Siron: ***"CERTI: Evolutions of the ONERA RTI Prototype"***, Fall Simulation Interoperability Workshop, Septembre 2002

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

[6] A.L.Wilson, R.M. Weatherly: ***"The aggregate level simulation protocol: an evolving system"***, WSC '94: Proceedings of the 26th conference on Winter simulation,1994.

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

[8] D.Jefferson: ***"Virtual Time"***, ACM Toplas, Vol.7, No 3, 404-425, july 1985.

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

[10] B.Samadi: ***"Distributed simulation algorithms andd performance analysis"***, Phd thesis, University of California, Los Angeles, 1985.

[11] F.Mattern: ***"Efficient algorithms for distributed snapshots and Global Virtual Time approximation"***, Journal of Parallel and Distributed Computing, 1993.

[12] T.McLean: ***"Hard Real-Time Simulations using HLA"***, Proceedings of the Simulation Interoperability Standards Organization (SISO) Simulation Interoperability Workshop, 2001.

[13] T.McLean, R.Fujimoto: ***"Predictable Time Management for Real-Time Distributed Simulation"***, Proceedings of the seventeenth workshop on Parallel and Distributed simulation, 2003.

[14] R. M. Fujimoto: ***"Lookahead in parallel discrete event simulation"***, In International Conference on Parallel Processing, Volume 3, pages 34–41, 1988.

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

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

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

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

[19] Concurrent Computer Corporation : ***"Real-Time Clock and Interrupt Module User's Guide"***, User Guide, August 2001.

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

[21] J.E.Hopcroft, J.D.Ullman: "***Introduction to Automata Theory, Languages and Computation"***, Addison-Wesley, 2001.

[22] G.Behrmann, A.David, K.G.Larsen: ***"A Tutorial on UPPAAL"***, White Paper, Department of

Computer Science, Aalborg University, Denmark, November 2004.

Author Biographies

PIERRE SIRON was graduated from a French High School for Engineers in Computer Science (ENSEEIHT) in 1980, and received his doctorate in 1984. He is currently a Research Engineer at ONERA and he works in parallel and distributed systems. He is leader of the CERTI Project. He is also Professor at the University of Toulouse, ISAE, and the head of the computer science program of the SUPAERO formation (French High School for Engineers in Aerospace Sciences).

ERIC NOULARD graduated from a French High School for Engineers in Computer Science (ENSEEIHT) in 1995 and received his PhD in computer science from Versailles University in 2000. After 7 years working in the Aerospace & Telecom domain for BT C&SI mostly for building high performance tests & validation systems he joined ONERA research center in Toulouse as Research Scientist. He works on distributed and real-time systems and his actively involved in the development of the CERTI and TSP Open Source projects.

JEAN-BAPTISTE CHAUDRON received the DEA in Artificial Intelligence (equivalent to fifth years of University studies) in Université Paul Sabatier, Toulouse, France. He pursues a PhD degree in computer science at ONERA/DTIM/SER, Toulouse, France. He is currently working on the extension of the High Level Architecture (HLA) towards the world of real-time.