Steffen Straßburger

Optimistic synchronization in the HLA 1516.1-2010 : interoperably challenged

URN: urn:nbn:de:gbv:ilm1-2015200198

Paper originated as a contribution to:

Fall Simulation Interoperability Workshop 2015 (SIW) : Orlando, Florida, USA, 31 August - 4 September 2015 / Simulation Interoperability Standards Organization (SISO)

Track: Services, Processes, Tools and Data (SVCS)

Paper: 15F-SIW-002

Optimistic Synchronization in the HLA 1516.1-2010: Interoperably Challenged

Steffen Strassburger
Department for Industrial Information Systems
Ilmenau University of Technology
Helmholtzplatz 3
98693 Ilmenau, GERMANY
steffen.strassburger@tu-ilmenau.de

Keywords:
Time Management, Optimistic Synchronization, flushQueueRequest.

ABSTRACT: *Time Management can be considered as one of the key achievements of the High Level Architecture for Modeling and Simulation (HLA). While HLA's time management is supposed to offer a unique support for heterogeneous time advancement schemes, its practical use is often limited to conservative time advancement (e.g. using services such as nextMessageRequest/nextMessageRequestAvailable) or time stepped time advancement (e.g. using services such as timeAdvanceRequest/timeAdvanceRequestAvailable). In this paper, we investigate HLA's capabilities for supporting optimistic time advancement and the interoperability between optimistic and conservative federates. The results are strikingly disappointing. While HLA had initially taken off with the noble vision of federations including both optimistic and conservative federates within a single federation execution, the current implementations of two leading RTI vendors fall short of achieving this objective. Neither do they enable the efficient execution of federations consisting of purely optimistically synchronized federates nor do they facilitate interoperability between optimistic and conservative federates. This paper documents the observed problems and discusses potential limitations in the IEEE HLA 1516.1-2010 specification and its interpretation by RTI vendors.*

1. Introduction

The HLA is a distributed simulation standard that intends to support heterogeneous time advancement schemes, including conservative and optimistic approaches. The basic idea of HLA Time Management (HLA TM) services is that HLA federates (i.e. individual simulations participating in an HLA based distributed simulation) have to request time advancement from the RTI. The RTI coordinates these requests and issues time advance grants according to the requests and guarantees it has.

Surveying the practical application of HLA shows that application of HLA TM, if used at all, has traditionally relied on services for conservative or time stepped synchronization, but did not often include optimistic synchronization services.

The infrequent usage of HLA-based optimistic synchronization for practical applications might be due to HLA not being a standard commonly frequented by the Parallel Discrete Event Simulation (PDES) community, at least if high performance and efficient execution are intended.

The few exceptions include Ferenci et al. [5], who investigated the options for federating different instances of Georgia Tech Time Warp (GTW) simulations and Vardanega and Maziero [16], who proposed the idea of a generic rollback manager for freeing optimistic HLA federates from some of the implementation overhead of optimistic synchronization.

When considering the application of HLA as interoperability standard for the connection of different commercial off the shelf simulation packages (CSP) [12], the choice of a performant synchronization scheme has a significant impact on execution speed and thus general acceptance.

Many interoperability problems encountered when connecting CSPs (see [10]) have an inherent zero Lookahead requirement. Since conservative protocols are known to have the worst performance under zero Lookahead conditions, the application of optimistic synchronization becomes appealing.

The general idea of optimistic synchronization is to allow simulations to process messages even if there is no guarantee that messages with a lower timestamp will not be received in their future. This "optimistic" execution of messages is based on the hope that causality violations, although possible, in fact will not or only sparsely occur. If an optimistically synchronized simulation receives a message that is in its logical past, it must take actions to reestablish causality. This is typically achieved by performing a rollback to a previously recorded state.

The only known research that investigated optimistic synchronization in the context of CSPs was conducted by Wang et al. [18]. Their work focused on ways of providing optimistic synchronization capabilities to a CSP in a manner that does not require major user involvement. While the application of optimistic synchronization to the domain of CSPs also builds the background for this paper, we here focus on the HLA related aspects of enabling optimistic synchronization and interoperability between optimistic and conservative federates. Aspects concerning the integration of optimistic synchronization into a CSP are reported in [11].

The remainder of this paper is structured as follows. Section 2 gives a brief introduction into HLA Time Management. Section 3 reviews the evolvement of those time management services that are supposed to enable optimistic synchronization over the different revisions of specification in the HLA. Section 4 introduces a small case study used to test optimistic synchronization and documents the results. Section 5 discusses potential changes needed in the HLA specification to prevent the problems observed in current RTI implementations.

2. Time Management in the HLA

The design and intentions of HLA Time Management were first described in [13] and [8]. Experiences with first implementations were published in [4].

As this paper obviously cannot discuss the entire design rational of HLA TM, this section intends to convey the essentials important for the remainder of this paper.

Time management in the HLA in general encompasses two aspects of federation execution, namely transportation services and time advancement services [13]. We here focus on time advancement services. They provide different primitives for federates to advance in logical time. Transportation services are equally important and provide different reliability and message ordering characteristics. For the purposes of this paper, we assume federates to use time-stamp ordered (TSO) and reliable message transport.

The general idea of HLA TM is that federates participating in HLA TM have to request the advancement of their logical time from the RTI. The RTI collects these requests and grants time advancement based on the requests and other guarantees (e.g. lookaheads) it is aware of.

Whether federates wish to participate in HLA TM is indicated by two logical switches named *time constrained* and *time regulating*. A *time constrained* federate is constrained by the logical time of other federates. A *time regulating* federate intends to participate in determining the logical time of other federates. Both switches are typically turned on for fully synchronized federates (only these are considered here).

Federates are further encouraged to indicate a Lookahead value to the RTI. Lookahead is a guarantee that a federate will not schedule any event with a time stamp less that the federate's current logical time plus the Lookahead value.

Initial versions of the HLA required a Lookahead value strictly greater than zero. This requirement was relaxed subsequently following a proposal made in [6].

HLA time advance services provide a means for the federate to request its advancement of logical time and to control the delivery of new messages to the federate. The following groups of time advance services are defined:

timeAdvanceRequest (TAR) / timeAdvanceRequestAvailable (TARA)

These services are intended for federates advancing its logical time in time steps. By invoking a *TAR(t)*, the federate is guaranteeing that it will not generate a TSO message at any time in the future with time stamp less than *t* plus that federate's Lookahead [13].

After invoking *TAR*, all messages eligible for delivery to the federate are passed to the federate by the RTI. A subsequent invocation of *timeAdvanceGrant (TAG)* by the RTI indicates to the federate that no additional TSO messages with time stamp less than or equal to *t* will be delivered in the future.

TARA(t) is the service flavor for zero Lookahead federates that want to be able to generate messages with a time stamp equal to the time returned by *TAG* as grant time.

nextMessageRequest (NMR) / nextMessageRequestAvailable (NMRA)[1]

These services provide support for conservative synchronization approaches. They are suitable for event driven simulations without rollback capabilities. An *NMR(t)* call can be interpreted as a request of the federate to advance the logical time of the federate to *t* or to deliver the next TSO message, provided that the message

[1] In earlier HLA versions, these services were called *nextEventRequest* and *nextEventRequestAvailable*.

has a time stamp no greater than *t*. A subsequent invocation of *TAG* by the RTI will return the time stamp of the TSO message delivered to the federate or *t* if no TSO messages were delivered. This in effect advances the federate's logical time to the value returned by *TAG*.

Once that *TAG* has been received, no subsequent TSO message will be delivered to the federate with a time stamp less than or equal to the federate's logical time.

It is worth noting that the parameter *t* passed in *NMR(t)* constitutes a conditional guarantee of the federate that it will not generate any new TSO messages with a time stamp less than or equal to *t*, unless it receives any TSO messages before the *TAG* call with a smaller time stamp than *t*.

NMRA(t) is the service flavor for zero Lookahead federates allowing a federate to still generate messages with a time stamp equal to the time returned in *TAG* as grant time.

flushQueueRequest (FQR)

FQR(t) is the service by which optimistic federates can request out-of-order delivery of TSO messages. *FQR(t)* releases all messages stored in the RTI's internal queues and delivers them to the federate invoking this service. *FQR(t)* can be considered as the central service for optimistic federates. Further details on its evolvement in the different HLA revisions are given in section 3.

Other services needed for optimistic federates include services to cancel sent messages (provided by the service pair *retract/requestRetraction*).

An essential additional requirement for optimistic federates is the need to be able to compute a lower bound on the logical time of any future rollback. This lower bound is called Global Virtual Time (GVT) and allows optimistic federates to free memory used for state checkpoints and message logs. In the HLA, the minimum of a federates LBTS[2] and the time stamp of messages in the RTI's local message queue provide the information necessary to determine GVT [7].

[2] LBTS can be defined as the "lower bound on the time stamp" of any subsequent message the RTI at a particular federate will receive from another federate [13].

3. Evolvement of FlushQueueRequest

3.1 HLA Time Management Design Document

The HLA Time Management Design Document (Version 1.0 from August 15, 1996) mentions both the optimistic execution among a collection of optimistic federates and federations including both optimistic and conservative federates as design goals for HLA TM [13]. HLA TM does not require all federates to support a rollback and recovery capability. Rather, optimistic messages are visible only to federates explicitly requesting to see them.

The service suggested for these purposes is *flushQueueRequest(t)*, or *FQR(t)* for short. This primitive releases all messages stored in the RTI's internal queues and delivers them to the federate invoking this service. All available TSO messages will be delivered, despite the fact that the RTI may not be able to guarantee that messages with a smaller time stamp could arrive later. The parameter *t* indicates that if the federate does not receive TSO messages with a time stamp smaller than *t*, then the federate's logical time can be advanced up to *t*.

It is important to note that invoking *FQR(t)* constitutes a conditional guarantee of the federate that it will not generate any new TSO messages with time stamp less than *t* plus the federate's lookahead if it does not receive any new TSO messages with time stamp less than *t*. In that regard, *t* has a similar importance as the *t* parameter in *NMRA(t)*.

The HLA TM Design Document [13] further suggests a dedicated *flushQueueGrant(t)* service, *FQG(t)* for short, that indicates that the *FQR(t)* service is completed. The time parameter of this call indicates that logical time for the federate has been advanced to this value and no additional TSO messages with a time stamp less than this value will be delivered in the future. This time parameter is defined "*as the lesser of LBTS and the time parameter of the Flush Queue Request that resulted in this call*" [13].

Please note that some discussion concerning this definition of the return value for *FQG(t)* is needed. In essence, it is defined as the minimum of LBTS and the *t* parameter passed in the preceding *FQR(t)* call. The importance of this return value is two-fold:

1) From the federate's point of view, the return value enables to determine GVT and perform fossil collection.
2) From the RTI's point of view, federate time is advanced to the return value, preventing the federate to send messages with a lower time stamp to the RTI.

In [17] it was shown, that the second implication is problematic to the federate, as the definition of the return value of *FQG* prevents a federate to respond to any message delivered after the *FQR(t)* invocation in a timely manner.

Beginning with HLA 1516-2000 this problem was solved and the minimum time stamp of any TSO message delivered in response to the *FQR(t)* call was added to the minimum determination expression defining the return value for *FQG*.

3.2 HLA Interface Specification Versions prior to 1.3
Due to time and space constraints, we refrain from discussing any versions of the HLA interface specification prior to version 1.3. For those interested, the HLA Programmers Guide for RTI 1.0.3 [14] implementing HLA Interface Specification Version 1.1 provides some historical reference.

3.3 HLA Interface Specification Version 1.3
The HLA Programmers Guide for RTI 1.3NG (implementing HLA Interface Specification Version 1.3) [15] defines *FQR* as follows:

"*When the flushQueueRequest() service is used, the federate's LRC will be eligible to release [...] all time-stamp ordered messages from the TSO queue. After all TSO messages that were in the queue at the time of the flushQueueRequest() invocation have been released, the federate will receive a timeAdvanceGrant() callback via the FederateAmbassador with time equal to LBTS or the time requested in the flushQueueRequest(), whichever is less*" [15, p. 6-4].

Parameter *t* passed in *FQR(t)* is defined as "*the maximum logical time to which to advance upon completion of the flush*" [15 (Appendix A), p. 5-10].

Essential for the return value of the following *TAG* call is the applied definition of LBTS: "*The LBTS specifies the time of the earliest possible time-stamp-ordered event the federate can receive. The LBTS is determined by looking at the earliest possible message that might be generated by all other regulating federates*" [15, p. 3-3].

Please note that there is a subtle distinction between LBTS and a quantity called "Minimum Next Event Time".

While "*LBTS is the greatest time-stamp such that it can be guaranteed that no time-stamp-ordered events will be subsequently generated in the federation with a lesser time-stamp*" [15 (Appendix A), p. 5-15] it may still be possible, "*that events with time stamps earlier than the LBTS may still be queued for time-stamp-ordered delivery to a federate; the LBTS merely indicates that no time-stamp-ordered events will be subsequently generated with an earlier time stamp*" [15 (Appendix A), p. 5-15].

Minimum Next Event Time includes these events and is defined as "*minimum time-stamp of all time-stamp ordered events that may be subsequently delivered to the federate*" [15 (Appendix A), p. 5-15].

Interestingly, there is an ambiguity in the HLA 1.3 NG Programmers Guide as to the return value of the *TAG* following *FQR*. While the definition given above talks about returning the minimum of LBTS and the *t* parameter passed in *FQR(t)*, the appendix A of the HLA 1.3 NG Programmers Guide specifies "*the minimum of the minimum-next-event time and the specified cutoff time*" [15 (Appendix A), p. 5-10] as return value of *TAG*.

Assuming that prior to the *TAG* call, all TSO events were delivered to the federate as mandated by *FQR*, LBTS and minimum next event time can be considered equivalent, though, healing this ambiguity.

In retrospective, the definitions discussed above are semantically equivalent to those from the HLA TM Design Document, with the single difference that the HLA Interface Specification 1.3 now does not use a dedicated *FQG* service for completing the *FQR*, but a unified *TAG* service, that completes all time advance services.

For further discussion, it is essential to note that

- *FQR* is clearly a service that shall advance logical federate time, and
- LBTS determination is necessary for determining the return value of the *TAG* following a *FQR*.

3.4 HLA Standard 1516.1-2000
The federate interface specification defined in HLA 1516.1-2000 [1] introduced several changes concerning HLA TM. While the definition of the intention of FQR remained unchanged ("*The FQR service shall request that all messages queued in the RTI that the joined federate will receive as TSO messages be delivered now*" [1, p. 143]), the specification of the resulting logical federate time was modified. The necessity of this modification was first discovered in [17] (see also discussion in section 3.1).

The resulting logical federate time (indicated by the return value of the subsequent *TAG* invocation) is now defined as the minimum of the

- logical time t passed in *FQR(t)*,
- the federate's GALT value (definition of GALT see below), and

- the smallest time stamp of all TSO messages delivered in response to the *FQR(t)* call.

Another apparent change in the HLA TM relates to the quantities LBTS and Minimum Next Event Time from HLA 1.3. Instead of these terms, HLA TM now introduces Greatest Available Logical Time (GALT) and Least Incoming Time Stamp (LITS).

GALT is defined as "*the greatest logical time to which the RTI guarantees it can grant an advance without having to wait for other joined federates to advance*" [1, p. 125].

A joined federate's LITS is "*the smallest time stamp that the joined federate could (but not necessarily will) receive in the future in a TSO message. A joined federate's LITS is calculated by the RTI and is based on the joined federate's GALT and any queued TSO messages that may later be received by the joined federate*" [1, p. 125].

Comparing the definitions, LITS is simply the new term for what was known as Minimum Next Event Time in HLA 1.3.

Although some authors see GALT just as well as a new term for LBTS [3], their definitions differ. Potential implications of these differing definitions remain to be discussed in due course of this paper.

Another apparently small addition to the description of *FQR* was made by introducing the sentence that an "*FQR can always be granted without waiting for other joined federates to advance*" [1, p. 143]. This sentence was not present the HLA 1.3 specification and seems to be of explanatory nature. However, the strict interpretation of this sentence can lead to severe interoperability problems. This sentence in essence encourages RTI developers to ignore the *t* parameter passed in *FQR(t)* when performing GALT computations. The ramifications of this will be discussed in section 4.

In retrospective, the HLA TM in HLA 1516.1-2000 corrected a mistake in the minimum definition of the return value of the *TAG* following a *FQR* call. At the same time, it introduced a new quantity named GALT to that minimum definition replacing the well-accepted LBTS. The specification unfortunately remains somewhat fuzzy on defining how GALT is to be computed ("*A joined federate's GALT is calculated by the RTI and is based on factors such as the logical time, lookahead, and requests to advance the logical time of time-regulating joined federates*" [1, p. 125]). The addition of the sentence described above furthers the fuzziness as it suggests that *t* in *FQR(t)* has no influence on GALT computation.

3.5 HLA Standard 1516.1-2010 ("HLA-Evolved")

HLA 1516.1-2010 [2] did not introduce any significant changes concerning *FQR*. It provided a small clarification towards the minimum determination discussed above. In addition, Annex E.8.1 now informs about the rational for the change made to the minimum determination introduced in HLA 1516-2000.

Further discussions in this paper are based on HLA 1516.1-2010 as the latest official version of the HLA standard.

4. Experiments

4.1 Experimental Setup

HLA's vision was to promote interoperability between federates using different time advancement schemes. To test the degree of fulfillment of this vision, we set up four distinct scenarios. Each scenario is a federation with two federates implementing a CSP interoperability reference model proposed in [10].

The scenario consists of two federates implementing a bounded buffer entity transfer problem of type IRM A.2 (Figure 1). Both federates exchange TSO interaction messages concerning entities to be transferred from federate 1 to federate 2 and concerning the content of queue Q2 in federate 2. It can be noted that the actual problem simulated is completely irrelevant for the further discussions.

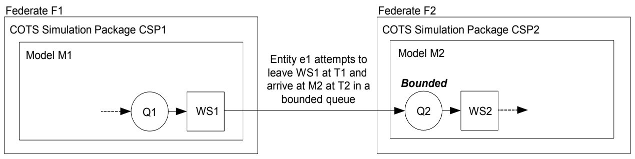
Federate F1
COTS Simulation Package CSP1
Model M1
Q1
WS1
Entity e1 attempts to leave WS1 at T1 and arrive at M2 at T2 in a bounded queue
Federate F2
COTS Simulation Package CSP2
Model M2
Bounded
Q2
WS2

Figure 1: Conceptual Model of IRM Type A.2 [10]

Both federates were implemented in SLX [9] (Version 2.3, Build EP 264). The commercial pRTI 5.0.0.0 (Build 1887) from Pitch was used as primary RTI software. The applied implementation of the SLX-HLA-Interface uses the HLA 1516.1-2010 ("HLA-Evolved") C++ API. The experiments were later repeated and verified against MÄK RTI 4.3 which exhibited the same behavior as observed with pRTI.

While the simulated behavior of both federates was kept constant, the time management characteristics were varied according to the design provided in table 1. Time management switches were set to time constrained and time regulating in all scenarios.

Table 1: Design of Experiments

	Federate 1	**Federate 2**
Scenario 1		
Synchronization	Optimistic (FQR)	Optimistic (FQR)
Lookahead	0	0
Scenario 2		
Synchronization	Optimistic (FQR)	Optimistic (FQR)
Lookahead	10	10
Scenario 3		
Synchronization	Optimistic (FQR)	Conservative (NMRA)
Lookahead	0	0
Scenario 4		
Synchronization	Optimistic (FQR)	Conservative (NMR)
Lookahead	10	10

Concerning the results, we observed the behavior of the time advancement in the federation. In specific, we recorded the return values obtained from the *TAG* services. Although we did not record wall clock time of each service call, Tables 2-5 provide a clear picture concerning the logical sequence of the individual calls. Please note that we do not display the *sendInteraction* and *receiveInteraction* calls in the tables. In cases where they had influence on the observed TAGs, they are mentioned in the textual descriptions.

4.2 Results

4.2.1 Scenario 1

In this scenario, both federates have zero Lookahead. They use FQR to advance through their simulation time. The results displayed in table 2 show that the return value of the *TAG* following an *FQR(t)* was always zero. This behavior was independent from the occurrence of any TSO message exchange.

Table 2: Excerpt from sequence of FQR/TAG calls for scenario 1

Sequence of calls	**Federate 1**	**Federate 2**
1	FQR(188.66) TAG (0.0)	FQR(86400.00) TAG (0.0)
2	FQR(377.33) TAG (0.0)	FQR(388.66) TAG (0.0)
3	FQR(754.66) TAG (0.0)	FQR(588.66) TAG (0.0)
…	…	…
n	FQR (86400.00) TAG (0.0)	FQR (86400.00) TAG (0.0)

The zero *TAG* values prevented any garbage collection required for optimistically synchronized federates. In essence, federate logical time and GALT were not advanced at all. Parameter *t* passed in *FQR(t)* did not have any influence on their determination.

4.2.2 Scenario 2

In this scenario, both federates have a Lookahead greater than zero (arbitrarily set to 10). They still used *FQR* to advance through their simulation time. The results displayed in table 3 show that the return value of the *TAGs* is now greater than zero (except for the very first *TAG* call), but still the return value has no correlation with the parameter *t* passed in *FQR(t)*. In essence, the return value of *TAG* now takes the guarantees into account that Lookahead provides.

Table 3: Excerpt from sequence of FQR/TAG calls for scenario 2

Sequence of calls	**Federate 1**	**Federate 2**
1	FQR(188.66) TAG (0.0)	FQR(86400.00) TAG (10.00)
2	FQR(377.33) TAG (20.0)	FQR(388.66) TAG (30.0)
3	FQR(754.66) TAG (40.0)	FQR(588.66) TAG (50.0)
…	…	…
n	FQR (86400.00) TAG (17000.00)	FQR (86400.00) TAG (17010.00)

Looking at line 1 in table 3, when federate 1 has received its *TAG(0.0)*, the RTI can safely advance federate 2's logical time to 10 (based on Lookahead information from federate 1). In the subsequent FQR call of federate 1, it can issue a *TAG(20.0)* based on federate 2's Lookahead of 10 (and so on).

While this behavior seems to provide some hope for optimistic federates to perform garbage collection, the observed way of determining the *TAG* value still ignores the conditional guarantee that the *t* parameter from *FQR(t)* provides.

4.2.3 Scenario 3

In this scenario, both federates have zero Lookahead again. Federate 1 is optimistic and uses FQR to advance through simulation time. Federate 2 is conservative and uses *NMRA* to advance simulation time. As can be seen from table 4, the *FQR* calls again always result in a *TAG(0.0)*, as in scenario 1.

The conservative federate 2 on the other hand was stuck after its first *NMRA* and waited for a *TAG*. This situation lasted until federate 1 resigned from the federation.

Only when federate 1 had resigned, would federate 2 receive a *TAG* to its very first *NMRA* call.

In case that federate 1 sent any TSO interaction messages to federate 2, the very first TSO message was delivered prior to that *TAG* call. The time stamp of the *TAG* call was then equal to the delivered TSO message. This case is displayed in table 4 – the *TAG(377.33)* indicates that an interaction message with that time stamp was received just prior to the *TAG* call (but only after federate 1 resigned).

The observed behavior in essence stalled any interoperability between optimistic and conservative federates at all. While federate 1 would simulate through its simulation time, federate 2 would only be allowed to continue simulation after federate 1 resigned. While Federate 2 would then still receive any TSO messages sent from federate 1, it was not able to react appropriately to these messages and deliver any feedback to federate 1.

Table 4: Excerpt from sequence of FQR/NMRA/TAG calls for scenario 3

Sequence of calls		**Federate 1 (F1)**	**Federate 2 (F2)**
F1	**F2**		
1	1	FQR(188.66) TAG (0.0)	NMRA (86400.00)
2		FQR(377.33) TAG (0.0)	
3		FQR(754.66) TAG (0.0)	
…		…	
n		FQR (86400.00) TAG (0.0)	
n+1		RFE[3]	
			TAG(377.33)
	2		NMRA(388.66) TAG (388.66)
	3		NMRA(588.66) TAG (588.66)
			…
	m		NMRA (86400.00) TAG(86400.00

With that, interoperability between conservative and optimistic federates with zero Lookahead has to be considered completely broken.

On side note, the observed behavior was independent from the time stamp passed in NMRA, e.g., issuing NMRA(600) instead of NMRA(86400) would result in the same sequence of calls as shown in table 4.

4.2.3 Scenario 4

In this scenario, both federates have a Lookahead greater than zero (arbitrarily set to 10), again. Federate 1 is optimistic and uses *FQR* to advance through simulation time. Federate 2 is conservative and uses *NMR* to advance simulation time.

The results here differ from scenario 3 only in that regard, that the optimistic federate 1 now receives non-zero *TAGs* that appear to somehow be based on the Lookahead values of both federates. The remainder of observations is identical to scenario 3.

Interoperability between conservative and optimistic federates with non-zero Lookahead has to be considered completely broken, too.

Table 5: Excerpt from sequence of FQR/NMR/TAG calls for scenario 4

Sequence of calls		Federate 1 (F1)	Federate 2 (F2)
F1	F2		
1	1	FQR(188.66) TAG (20.0)	NMR (86400.00)
2		FQR(377.33) TAG (40.0)	
3		FQR(754.66) TAG (60.0)	
…		…	
n		FQR (86400.00) TAG (17000.00)	
n+1		RFE[3]	
			TAG(377.33)
	2		NMR(388.66) TAG (388.66)
	3		NMR(588.66) TAG (588.66)
	…		…
	m		NMR (86400.00) TAG(86400.00

Comparing scenarios 2 and 4, the observed return value of the *TAG* in scenario 4 seems to be even more erratic and inexplicable. While the TAGs in scenario 2 were consistently based on Lookahead provided guarantees only, this is not the case in scenario 4.

[3] RFE = resignFederationExecution

4.3 Discussion

From the documented scenarios it becomes obvious that the *t* parameter passed in *FQR(t)* is not taken into account when determining the return value of the resulting *TAG* call.

While the HLA 1516.1-2010 actually states that GALT computation shall consider factors such as "*requests to advance the logical time"*, RTI implementations seem not to consider *FQR(t)* as such a request.

While this is a nuisance for optimistic federates (as they cannot comfortably determine GVT), this becomes a show-stopper for federations involving optimistic and conservative federates.

While the described observations seem to be obviously different from what HLA Time Management intended, feedback from Pitch Priority Support is that they consider the behavior as a correct interpretation of the HLA standard. The main issue at hand here is, which influence the parameter *t* passed in *FQR(t)* shall have on GALT calculation, as GALT is fundamental for the return value of the *TAG* following a *FQR(t)*.

There are two aspects in the *FQR* definition [2, p. 171f] that support the interpretation embraced by Pitch:

1) The *FQR* definition mentions that "A *FQR service can always be granted without waiting for other joined federates to advance."* [2, p. 171]. This basically can be interpreted as an invitation not to perform any GALT calculations when FQR is called.

2) The return value of the resulting *TAG* takes reference on GALT. GALT however, is defined differently from LBTS as "*the greatest logical time to which the RTI guarantees it can grant an advance without having to wait for other joined federates to advance"* [2, p. 153]. While the LBTS definition (see 3.3) seems comparable ("*LBTS is the greatest time-stamp such that it can be guaranteed that no time-stamp ordered events will be subsequently generated in the federation with a lesser time-stamp"),* it does not contain the latter part of the GALT definition ("*without having to wait for other joined federates to advance*"). The interpretation of the GALT definition could therefore be that GALT, once determined, is locally correct and independent from other federates. The LBTS definition on the other hand would likely have to be interpreted in such a way, that whenever LBTS is referred to, a distributed snapshot calculation of LBTS of the entire federation is needed.

In essence, the strict interpretation of the HLA1516.1-2010 could be that *FQR(t)* can trigger a *TAG* immediately based on locally available GALT information - without starting a new GALT computation and thus ignoring *t*.

With that interpretation, the design goals of HLA TM cannot be achieved. If that interpretation sustains, a change to the HLA specification is required.

5. Summary and Recommendation

This article has investigated HLA's support for interoperability between federates using different time advancement schemes. Investigations were based on the (at the time of writing) current HLA 1516.1-2010 ("HLA-Evolved") specification and the RTI implementations of two leading RTI vendors implementing this standard.

In specific, interoperability between purely optimistic federates and interoperability involving both optimistic and conservative federates was tested.

The results were strikingly disappointing. Interoperability between purely optimistic federates was handicapped, as the determination of global virtual time (GVT) was severely hampered (Lookahead greater than zero) or even completely impossible (Lookahead equal to zero).

Interoperability between a conservative and an optimistic federate was completely halted, as the conservative federate would block until the optimistic federate would resign.

The reasons for this lack of interoperability seem to lie in the way HLA definitions of the flushQueueRequest service and the Greatest Available Logical Time (GALT) are interpreted by RTI vendors.

To prevent the interpretation currently embraced by Pitch, we suggest the following modifications to the HLA standard.

Recommendation 1: Remove the sentence "*A Flush Queue Request service can always be granted without waiting for other joined federates to advance.*" [2, p. 171] from the definition of the Flush Queue Request Service.

Recommendation 2: Clarify the required treatment of conditional guarantees expressed via the time parameter *t* of *NMR(t)*/*NMRA(t)* and *FQR(t)* and their necessary influence on GALT computations ([2] - Section 8.1.5 Time-constrained joined federates).

Recommendation 3: Clarify on which occasions a new GALT computation shall be required and which service invocations shall initiate it. This clarification could enhance the service descriptions of all time advancement services (e.g., "*Each invocation of FQR made by a time regulating federate shall trigger a new GALT computation.*"), or be put into section E.8 of [2].

6. References

[1] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). Federate Interface Specification*. Institute of Electrical and Electronics Engineers, New York, IEEE Std 1516.1-2000.

[2] *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). Federate Interface Specification*. Institute of Electrical and Electronics Engineers, New York, IEEE Std 1516.1-2010.

[3] Buquan Liu, Yiping Yao, and Huaimin Wang. 2007. An Efficient Algorithm in the HLA Time Management. In *Proceedings of the 2007 Winter Simulation Conference. December 9 - 12, 2007, Washington, DC, U.S.A*. ACM, IEEE, New York, NY, 585–593.

[4] Carothers, C. D., Fujimoto, R. M., Weatherly, R. M., and Wilson, A. L. 1997. Design and Implementation of HLA Time Management in the RTI Version F.0. In *Proceedings of the 1997 Winter Simulation Conference. Atlanta, GA, 7-10 December 1997*. ACM, IEEE, Piscataway, NJ, 373–380.

[5] Ferenci, S. L., Perumalla, K. S., and Fujimoto, R. M. 2000. An Approach for Federating Parallel Simulators. In *14th Workshop on Parallel and Distributed Simulation (PADS 2000)*, 63–70. DOI=10.1109/PADS.2000.847145.

[6] Fujimoto, R. M. 1997. Zero Lookahead and Repeatability in the High Level Architecture. In *1997 Spring Simulation Interoperability Workshop*.

[7] Fujimoto, R. M. 1998. Time Management in the High Level Architecture. *SIMULATION* 71, 6, 388–400.

[8] Fujimoto, R. M. and Weatherly, R. M. 1996. Time Management in the DoD High Level Architecture. *SIGSIM Simul. Dig.* 26, 1, 60–67.

[9] Henriksen, J. O. 1999. SLX - The X is for eXtensibility. In *Proceedings of the 1999 Winter Simulation Conference*, 167–175.

[10] Simulation Interoperability Standards Organization. 2010. *Standard for Commercial-off-the-shelf Simulation Package Interoperability Reference Models (SISO-STD-006-2010)*. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30829.

[11] Strassburger, S. HLA-based Optimistic Synchronization with SLX. In *Proceedings of the 2015 Winter Simulation Conference*, L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal and M. D. Rossetti, Eds.

[12] Taylor, S. J. E., Turner, S. J., Strassburger, S., and Mustafee, N. 2012. Bridging the Gap: A Standards-Based Approach to OR/MS Distributed Simulation. *ACM Trans. Model. Comput. Simul.* 22, 4, 1–23.

[13] U.S. Department of Defense. 1996. *HLA Time Management Design Document. Version 1.0.* August 15, 1996.

[14] U.S. Department of Defense. 1997. *High Level Architecture Run-Time Infrastructure Programmer's Guide (Version 1.0 Release 3)*.

[15] U.S. Department of Defense. 2002. *RTI 1.3-Next Generation Programmer's Guide (Version 5)*.

[16] Vardanega, F. and Maziero, C. 2000. A Generic Rollback Manager for Optimistic HLA Simulations. In *4th IEEE International Workshop on Distributed Simulation and Real Time Applications (DS-RT 2000)*, 79–85.

[17] Wang, X., Turner, S. J., Low, Malcolm Y. H., and Gan, B. P. 2004. Optimistic Synchronization in HLA Based Distributed Simulation. In *18th Workshop on Parallel and Distributed Simulation*, 123–130. DOI=10.1109/PADS.2004.1301293.

[18] Wang, X., Turner, S. J., Low, Malcolm Y. H., and Gan, B. P. 2005. Optimistic Synchronization in HLA-Based Distributed Simulation. *SIMULATION* 81, 4, 279–291.

Author Biography

STEFFEN STRASSBUGER is a professor at the Ilmenau University of Technology and head of the Department for Industrial Information Systems. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Doctoral and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He is further an associate editor of the Journal of Simulation. His research interests include distributed simulation, automatic simulation model generation, and general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Support Group. His web page can be found via www.tu-ilmenau.de/wi1. His email is steffen.strassburger@tu-ilmenau.de.