A Discussion of Time Management Concepts and Time Constraint Equations for Multi-Rate Federation Executions

Edwin Z. Crues, Ph.D.
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
edwin.z.crues@nasa.gov

Daniel E. Dexter
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
daniel.e.dexter@nasa.gov

Keywords: HLA, Time, Time Management, Time Synchronization, SpaceFOM

ABSTRACT: *The High Level Architecture (HLA) is a simulation interoperability standard developed by the Simulation Interoperability Standards Organization (SISO) and published as the international standard IEEE 1516-2010 by the Institute for Electrical and Electronics Engineers (IEEE). HLA is a widely used standard for the development and execution of collaborative distributed simulations. HLA provides a number of Management Services to simulation developers: Federation, Declaration, Object, Ownership, Data Distribution, and Time. Of those services, Time Management Services is probably one of the least understood and least used. However, Time Management Services are critical to technical simulations like those created for space systems using the Space Reference Federation Object Model (SpaceFOM). Time Management can be used to ensure data coherence and execution repeatability in distributed simulations. When combined with real time execution policies, Time Management is being used to support real time execution of mixed software and hardware in the loop integration, verification, and validation simulations for active space systems development. This paper starts by providing an overview of the HLA Time Management Services. This provides the background to discuss the challenges associated with Time Management and its use, starting with simple common rate frame scheduled simulations, then simple multi-rate simulations, and ending with complex mixed rate simulations. The authors then formulate the significant time constraint relationships between identified frame scheduling parameters. The intent of the paper is to provide a concise discussion of how to use Time Management in both simple cases and in more complex mixed frame rate federation executions.*

1 Introduction

Modeling and simulations (M&S) has a long, historic, and documented use across many areas of human experience and particularly in areas associated with technology and technology development. This is particularly true for human space system development, which is the professional background of the authors of this paper. With the advent of network connected digital computational systems, simulation technologies have evolved to take advantage of the possibilities that distribution and parallelism can provide. However, like many technological advancements, along with the possibilities of distributed simulation technologies come limitations, restrictions, and constraints.

Over the years, a number of distributed simulation technologies have been developed. One of the more widely used and well developed collaborative distributed simulation standards is the High Level Architecture (HLA). HLA provides a rich set of capabilities subject to documented limitations, restrictions, and constraints. These are enumerated in the *Framework and Rules* volume of the standard [1]. The HLA framework is formulated into a collection of constituent Management Services: Federation, Declaration, Object, Ownership, Data Distribution, and Time. Of those services, Time Management Services is probably one of the least understood and least

*U.S. Government work not protected by U.S. copyright used. Regardless, Time Management Services are critical to formulating HLA-based distributed simulations that maintain coherent data exchange and have repeatable execution results. Real time hardware-in-the-loop (HWITL) and human-on-the-loop (HITL) requirements further complicate the use of Time Management Services. However, these are important requirements of many simulations supporting human spaceflight systems and critical to real time integrated verification and test simulations.

The Space Reference Federation Object Model (SpaceFOM) standard was specifically designed to support the specific needs of human spaceflight simulations [2]. SpaceFOM supports simulation executions that are unrestricted with respect to real time and also supports real time paced simulations. SpaceFOM-based simulation is being used to support real time execution of mixed software and hardware in the loop integration, verification, and validation simulations for active space systems development. As a result, HLA Time Management Services provide an essential capability to SpaceFOM-based simulations.

This paper focuses on a particular set of limitations, restrictions, and constraints; specifically, those associates with the modeling of time, managing the progression of time, and the associated synchronization challenges. Section 2 of this paper will provide an overview of HLA and the services provided. In particular, this section will focus on Time Management Services and present the technology basis, principal concepts, and working terminology for the paper. Section 3 will provides an overview of the SpaceFOM standard and discuss the importance of Time Management Services in its development and use. Section 4 will discuss the principal challenges associated with HLA Time Management and its use in mixed frame rate federation executions. Section 5 presents a collection of constraint equations that can be used to check timing consistency for individual federates and across a federation execution. The paper concludes with discussions of what should be the principal take aways from reading the paper in Section 6 and outlines for potential future discussion topics and work in Section 7.

2 High Level Architecture

HLA is developed and maintained by the Simulation Interoperability Standards Organization (SISO) and is published by the Institute for Electrical and Electronics Engineers (IEEE) as standard IEEE-1516 and titled: *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA).* The current version of the HLA standard is IEEE 1516-2010, also know as HLA Evolved.[1] The HLA standard is published in 3 volumes: Framework and Rules [1]; Federate Interface Specification [3]; and Object Model Template (OMT) Specification [4]. HLA provides a well developed, tested, maintained, and widely used distributed simulation framework with a number of reliable commercially available implementations.

The HLA framework is formulated into a collection of constituent **Management Services**: **Federation**, **Declaration**, **Object**, **Ownership**, **Data Distribution**, and **Time**. These services are defined in detail in the *Federate Interface Specification* document [3]; but, to help the reader, brief definitions are provided here:

Federation Management Services: refer to the creation, dynamic control, modification, and deletion of a federation execution.

Declaration Management Services: refer to services to declare the intent to generate information.

Object Management Services: deal with the registration, modification, and deletion of object instances and the sending and receipt of interactions.

Ownership Management Services: is used by joined federates and the RTI to transfer ownership of instance attributes among joined federates.

Data Distribution Management Services: used by joined federates to reduce both the transmission and the reception of irrelevant data.

Time Management Services: provide a federation execution with the means to order the delivery of messages throughout the federation execution.

[1]A new HLA Version 4 is expected to be released even before this paper is published.

Of these services, Time Management Services are probably one of the least understood and least used, other than in the simple required advancement of the HLA logical timeline. Regardless, Time Management Services are critical to formulating HLA-based distributed simulations that maintain coherent data exchange and have repeatable execution results.

2.1 Time Management Services

Within the limitation of this paper, we will cover a few key concepts associated with HLA Time Management. However, for more details and a better understanding of HLA Time Management, the reader is referred to the HLA standard itself [1]. This paper will focus on a few details of HLA Time Management that are critical to its use for time synchronized, real time, and repeatable HITL and HWITL distributed simulations.

HLA Time Management services provide the means for joined federates in an HLA-based federation execution to operate with a logical shared concept of time in the form of a distributed virtual clock. These services provide the infrastructure required to assure a causal ordering of time-based events. This is a requirement for time-based physics-based technical simulations like those used to support human spaceflight activities. The HLA standard uses a concept called HLA Logical Time (HLT). Each individual federate maintains its own local concept of HLT. Through the HLA Time Management Services, these federate specific concepts of time progression can be linked together to form a federation execution wide progression of time.

2.1.1 HLA Logical Time Representations

Note that HLA does not have a 'standardized representation' for time. However, there are two principal standard types that are used: 64-bit integer time and 64-bit float time. These each have separate API interfaces. This paper will proceed using the **HLAInteger64Time** time representation. It is important to note that this time representation does not have any direct association with a 'continuous' time scale. The mapping between HLT and federate time scale is defined by the federation execution. The SpaceFOM does specify a mapping which is discussed in Section 3.1.

2.1.2 Time Advance Request (TAR) versus Time Advance Grant (TAG)

At the core of HLA's Time Management approach is HLT. HLT is used in conjunction with a collection of time advance request and grant mechanisms. In this paper, we will discuss only the two generally used methods of Time Advance Request (TAR) and Time Advance Grant (TAG). The interval between a TAR and a TAG is referred to as Time Advancing state.

TAR is a federate service call used to request the HLA Runtime Infrastructure (RTI) to advance HLT to a desired time. The requested time must be greater than or equal to the current HLT plus a time buffering parameter called the Lookahead (La) time step. Simply stated, La is needed to provide flexibility in the coordinated advancement of time. Further discussion of La and its role in HLA time advancement, while interesting and relevant, is beyond the scope of the paper.

TAG is a coordinated service callback from the RTI that tells a federate that a coordinated advancement in time has happened for the entire federation execution. Even though the TAG is handled through a callback mechanism, the local federate must wait to advance the federate's local concept of time until the start of the corresponding simulation frame for the granted time. If the Next Message Request, Next Message Request Available, or Flush Queue Request services are used the granted time may not be to the HLT requested by a federate; as a result, the current HLT should always be checked before advancing the time of the local federate. The local federate must wait until a TAG for a time greater than or equal to the desired time indicated by the RTI. Only then should it advance the federate's local concept of time.

2.1.3 Time Constrained versus Time Regulating

In addition to TAR and TAG, HLA Time has two important time management concepts associated with constraining the advancement of time and regulating the advancement of time. These concepts are not only important in coordinating the advancement of time across a federation execution, they also play a role in the way certain data is sent and received for federates in a federation execution. Again these data delivery implications are beyond the scope of this paper. Here, we will concentrate on the time advancement implications of Time

Constrained versus Time Regulating federates in an HLA Time Managed federation execution. The HLA Time Management services provide the ability for a federate to participate as a Time Constrained federate, as a Time Regulating federate, neither, or both.

A Time Regulating federate assumes a responsibility in a federation execution that requires it to advance time using the TAR service calls. A time managed federation execution can only advance HLT once all Time Regulating federates have issued a TAR.

A Time Constrained federate is constrained in its ability to advance its local HLT subject to a TAG. A Time Constrained federate will only receive a TAG for a specified time once all time regulating federates have issued a TAR for that specified time.

It is important to note that HLT is not a continuous or uniform time scale. As a result of the underlying TAR and TAG relationship, HLT will advance in discreet steps from one HLT to the next depending on the TAR and TAG sequencing between participating time managed federates. This is important when HLA Time Management is used in real time simulations.

3 SpaceFOM

The SpaceFOM defines a collection of HLA-compliant data constructs, modeling standards, and execution control process standards that support interoperability between simulations in the space domain [2, 5]. It is designed to link simulations of discrete physical entities into distributed collaborative simulations of complex space related systems. The SpaceFOM provides the following:

- Definitions of specific roles and responsibilities of federates within a federation execution;
- Definitions of common datatypes useful in the space domain;
- Definitions of common time lines and time scales needed for time homogeneity;
- Definitions for specific time-stepped focused time management approaches;
- A flexible positioning system, using reference frames for locating arbitrary bodies in space;
- A naming convention for operational reference frames;
- Support for physical entities (e.g., space vehicles and astronauts);
- Support for physical interfaces (e.g., docking ports and sensor locations);
- A synchronized execution control strategy and framework;
- Rules defining the requirements for SpaceFOM compliance;
- A core base set of FOM modules needed for a SpaceFOM-compliant federation execution.

3.1 Time Lines

The SpaceFOM defines the following 6 distinct time lines associated with a SpaceFOM-based federation execution:

Physical Time (PT): the non-spatial dimension associated with our spacetime continuum in which events are ordered in irreversible succession from the past to the present to the future.

Computer Clock Time (CCT): the representation of time maintained and managed by the computer usually using some form of oscillator counting oscillations from a known epoch. Mapped into an approximation of PT.

Simulation Elapsed Time (SET): the time measure associated with an individual simulation starting at zero and advancing monotonically in quantifiable steps.

Simulation Scenario Time (SST): a model within a simulation that associates the Simulation Elapsed Time with a representation of the problem's Physical Time.

HLA Logical Time (HLT): the time line used by HLA to order messages, regulate execution time advance (TAR & TAG), and enable deterministic behavior in distributed simulation.

Federation Scenario Time (FST): a time associated with the physical systems being modeled in the participating federates in the federation execution.

These time lines are important to understanding how the progression of simulation time is managed between various federates in a SpaceFOM-based federation execution. These concepts are the working variables for the ensuing time synchronization and constraint discussions.

Remember from Section 2.1.1 that HLA does not have a defined time representation. This poses a problem when tying discreet HLT to the SST described above. The SpaceFOM address this issue by specifying a mapping. Currently, the SpaceFOM uses an **HLAInteger64Time** time representation with a mapping of one 1 HLA integer time tick (epsilon value) to 1 microsecond of SST. This results in the association that a 1.0 second time interval in SST is equal to 1,000,000 in HLT. Note that this limits the effective resolution of time to 1 microsecond.[2]

3.2 Roles and Responsibilities

The SpaceFOM standard defines three key roles in any SpaceFOM-compliant federation execution: Master, Pacing, and Root Reference Frame Publisher. The Master federate manages the overall execution of the federation execution. This includes managing mode transitions: Run, Freeze, and Shutdown. The Master federate also manages the time synchronization by providing the scenario time epoch, mode transition times, central timing equipment times, and the least common time step for the current federation execution. The Pacing federate provides the time regulating clock for the overall federation execution. This can be faster than real time or regulated to real time. The Pacing federate uses the native HLA Time Management services to regulate the advance of HLT in coordination with the Simulation Scenario Time. The Root Reference Frame Publisher provides the federation execution with the name of the root reference frame for the federation execution reference frame tree. These roles can be split between participating federates or hosted within a single federate.

3.3 Time Management Options

The SpaceFOM standard supports time stepped simulation based on fixed uniform time steps. Specifically, it relies on the use of the HLA Time Management services to coordinate the time-based progression of participating federates using uniform fixed increments of HLT. This includes support for Time Constrained and Time Regulating federates.

SpaceFOM also supports the ability to constrain the rate of time advance in a SpaceFOM-based federation execution. The process of constraining the rate of time progression is called Pacing. For non-pacing federation execution, no Time Regulating federate in the federation execution is constrained in its advancement of time other than in its ability to run as fast as possible. Note that this does not mean that the federation execution will run that fast. In fact, the federation execution will run only as fast as the slowest Time Regulating federate. However, it will not be regulated or paced by any real time clock on any computer.

SpaceFOM also supports paced federation execution. In general, paced execution refers to a federation execution that is regulated and constrained by the progression of a computer clock in the Pacing federate. However, there are a number of variations in how a federation execution can manage the progression of CCT and the enforcement of time schedule boundaries. SpaceFOM categorizes real time execution types into the following hierarchy:

1. No Pacing (as-fast-as-possible)
2. Pacing (regulated by a clock)

[2]This can be a limiting restriction when dealing with some commonly used cycle time frame scheduling values. For instance a 128 Hz cycle frequency has a 0.0078125 second cycle time. This exceeds the 1 microsecond resolution. Future versions of SpaceFOM may allow the use of alternate time resolutions.

a. Scaled Pacing

b. Real-time Pacing

   i. Strict/Conservative Real-time (no frame overruns)

   ii. Elastic Real-time (catch-up on overruns)

      1). Limited Overruns

      2). Unlimited Overruns

For more information on these SpaceFOM topics the reader is encouraged to reference the SpaceFOM standard's detailed discussions [2].

For the remainder of the paper, we will assume a federation execution that is running with a real time pacing federate (2.b above). We will assume that all frame scheduling boundaries are met, not consider how overrun events are handled. Finally, we will assume that all federates are Time Constrained and Time Regulating.

4 The Challenges of Time Synchronization

Having laid the HLA and SpaceFOM groundwork in the preceding sections, we can now focus on the challenges associated with coordinating time synchronized federation executions.

4.1 Nomenclature

The following nomenclature will be used in the ensuing discussion and the formulation of the constraint equations.

dt_i	Principal time step frame for federate i.
EO_i	Executive overhead for federate i.
FM_i	Frame margin for federate i.
FP_i	Principal frame processing time for federate i.
La_i	Lookahead value for federate i.
$LCM()$	Least Common Multiple operation.
$LCTS$	Least Common Time Step of all federates.
N	Number of Time Constrained and/or Time Regulating federates in a federation execution.
RT	Real Time frame time for pacing federate.

4.2 Basic Concepts

A discussion of time synchronization requires the definition of some key concepts and parameters.

4.2.1 Basic Simulation Execution Frame Concepts

Figure 1 shows a simplified case where the principal federate time step (dt) is the same as the real time frame (RT), in this case 0.025 seconds or 40 cycles per second (Hz). This helps to show the relationship between the frame processing time (FP), executive overhead (EO), and principal time step (dt).

$$dt_i = FP_i + EO_i + FM_i \tag{1}$$

We can solve for the real time frame margin by rearranging Equation 1 as follows:

$$FM_i = dt_i - (FP_i + EO_i) \tag{2}$$

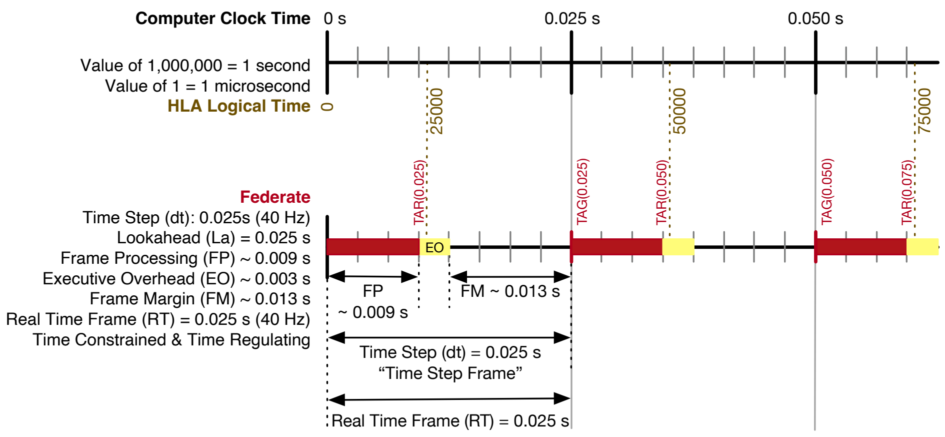

Computer Clock Time
0 s
0.025 s
0.050 s
Value of 1,000,000 = 1 second
Value of 1 = 1 microsecond
HLA Logical Time
0
25000
50000
75000
Federate
Time Step (dt): 0.025s (40 Hz)
Lookahead (La) = 0.025 s
Frame Processing (FP) ~ 0.009 s
Executive Overhead (EO) ~ 0.003 s
Frame Margin (FM) ~ 0.013 s
Real Time Frame (RT) = 0.025 s (40 Hz)
Time Constrained & Time Regulating
TAR(0.025)
TAG(0.025)
TAR(0.050)
TAG(0.050)
TAR(0.075)
EO
FP
~ 0.009 s
FM ~ 0.013 s
Time Step (dt) = 0.025 s
"Time Step Frame"
Real Time Frame (RT) = 0.025 s

Figure 1: **HLA Time Management Basic Frame Concepts.**

In this case, the computational work done during the frame (FP) takes approximately 0.009 seconds. The simulation executive takes an additional approximately 0.003 seconds. Since the real time frame is 0.025 seconds, this leaves approximately 0.013 seconds of remaining margin in the real time frame. Note that in order to maintain a real time execution $FM_i > 0$.

Note that the HLT values do not line up with the corresponding values on the CCT timeline. This is because, at present, there is no connection between the CCT timeline and the federate execution time line. HLT is advanced only when all present Time Regulating federates have issued TAR for the next desired time value, in this case to a value of 25000.[3] A TAR to 25000 occurs somewhere near the end of the principal frame processing sequence. HLT can then advance to 25000. This occurs well before the corresponding real time frame value of 0.025 seconds on the CCT timeline. HLT and CCT are only synchronized at the CCT real time frame boundary. Because the real time frame and the federate time step are 0.025 seconds, the federate's next time step starts on the real time boundary of 0.025 seconds.

4.2.2 The Effects of Real Time

What happens if the federate time step is not the same as the real time frame?

Figure 2 shows a slightly more complex case where the real time frame is twice the federate time step. In this case, the federate time step is still 0.025 seconds (40 Hz). However, the real time frame is now 0.050 seconds (20 Hz). The real time frame is twice the federate time step.[4]

Now, not only does the HLT steps not line up with the CCT timeline, not all the federate time steps line up with the CCT timeline. Specifically note that the second time step starting at 0.025 seconds begins immediately after the completion of the first time step starting at 0.0 seconds. This is because the federate will not check for real time until the real time frame (0.050 seconds). Until then, all jobs run as fast as possible. This means that the second time step will start at approximately 0.012 seconds on the CCT timeline after the last frame of the preceding step completes along with some additional Executive Overhead (EO).

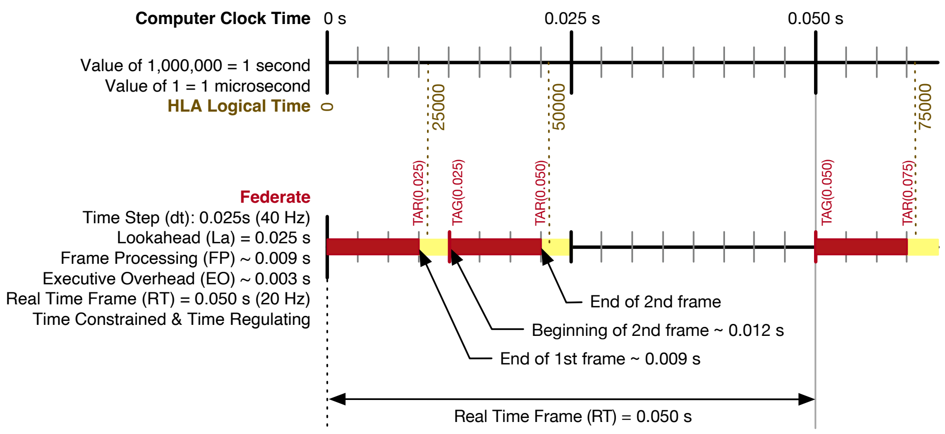
After the completion of the second frame, the federate will wait until the CCT timeline reaches 0.050 seconds before scheduling any other jobs to run. At this point, the HLT and CCT timelines resynchronize at 50000 and 0.050 seconds respectively. The cycles repeat every 0.050 seconds.

4.2.3 Simple Matched Rate Multi-Federate Case

What happens with an additional federate with the same frame scheduling?

[3]Note that an HLT integer value of 25000 corresponds to 0.025 seconds.

[4]For reasons we will not go into in this paper, it is best if there is an integer multiple relationship between the federate time step and the real time frame.



Computer Clock Time
0 s
0.025 s
0.050 s
Value of 1,000,000 = 1 second
Value of 1 = 1 microsecond
HLA Logical Time
0
25000
50000
75000
Federate
Time Step (dt): 0.025s (40 Hz)
Lookahead (La) = 0.025 s
Frame Processing (FP) ~ 0.009 s
Executive Overhead (EO) ~ 0.003 s
Real Time Frame (RT) = 0.050 s (20 Hz)
Time Constrained & Time Regulating
TAR(0.025)
TAG(0.025)
TAR(0.050)
TAG(0.050)
TAR(0.075)
End of 2nd frame
Beginning of 2nd frame ~ 0.012 s
End of 1st frame ~ 0.009 s
Real Time Frame (RT) = 0.050 s

Figure 2: **HLA Time Management Realtime Effects.**

Figure 3 shows the case where there are two federates in the federation execution with matching frame scheduling: Master/Pacing Federate (MPF) and Federate 2 (F2).[5] Both federates have a federate time step of 0.025 seconds (40 Hz). The MPF has a real time frame of 0.025 seconds (40 Hz). Federate 2 does not have a real time frame and runs as fast as possible.[6]

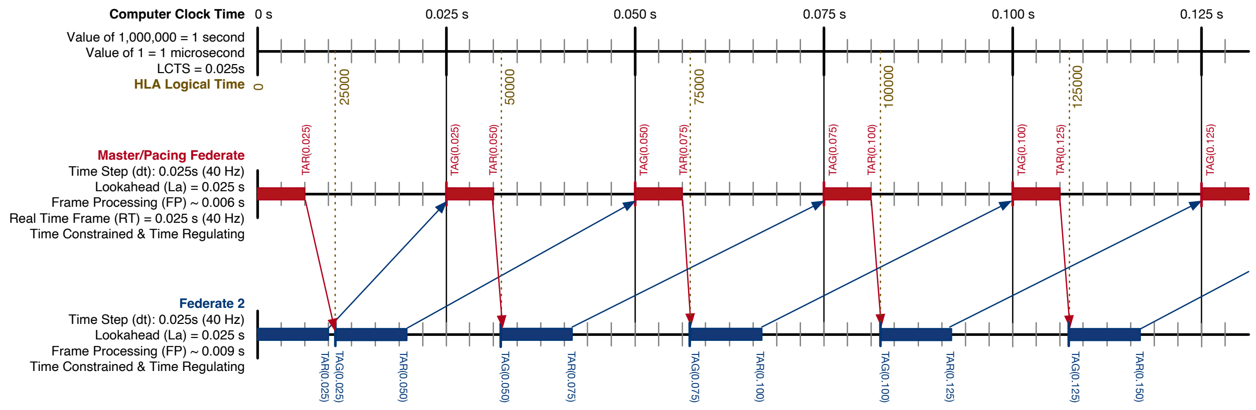


Figure 3: **HLA Time Management Simple Matched Timing.**

In this example, two federates are 'managing' time using the HLA TAR/TAG APIs. These federates have matching frame schedules (0.025 seconds) but differing Frame Processing (FP) times, ~0.006 and ~0.009 seconds. Note that each federate has positive margin (FM). However, the job scheduling for each federate with respect to the CCT timeline are very different.

The MPF assumes the Pacing role for this federation execution. The real time frame is 0.025 seconds. This scheduling for the MPF looks very much like that in Figure 1. Federate F2 is not a real time federate but does have a frame time step (dt) of 0.025 seconds. For the first frame, the F2 frame scheduling looks more like Figure 2. However, it looks more like Figure 1 for all subsequent frames but starting earlier than the real time frame would suggest.

Why is that?

In this case, federate F2 itself does not have a real time clock but is 'constrained' to run real time through the HLA Time Management APIs, specifically TAG, and the regulating effects of the MPF. Remember that

[5]To simplify the diagrams, from this point on, the Executive Overhead (EO) will be absorbed in the overall processing time and not shown independently.

[6]There should only be one real time source for a time synchronized, time constrained, time regulating federation execution. Again, it is beyond the scope of this paper to explain why.

HLT cannot advance until all participating regulating federates have made a TAR to the desired HLT. In the first frame, this comes at ∼0.009 seconds after F2 makes its TAR to 25000 (0.025 s). Then, after some RTI processing, a TAG will succeed for an HLT up to and including 25000. This 'releases' federate F2 to begin its second frame at ∼0.012 seconds. The MPF second frame could also start at the same time but is held back waiting for the real time CCT to reach 0.025 seconds. Since the second frame for federate F2 has already begun and it has already issued a TAR for 50000, then frame 3 for F2 is only waiting for MPF frame 2 to complete and issue a TAR for 50000. The pattern repeats for every frame after that.

Also note the arrows in the figure. These indicate a conceptual attribute update from one federate to the other. The attribute updates are sent at the end of the computational frame for each respective federate. By the rules associated with HLA Time Management services, attribute updates can only be 'sent' with HLT values greater than or equal to the currently granted HLT plus the Lookahead (La) time. In this example, the federates both have an La of 25000 (0.025 s) that match the federate time steps (dt) of 0.025 seconds. This results in attribute updates from one frame not being 'received' until the TAG for the next frame.

4.2.4 Simple Multi-Rate Multi-Federate Case

What if the time steps (dt) for federates do not match?

Having covered some basic concepts and a simple multi-federate example, we can begin to look at a more complex case where participating federates have different federate time steps. Figure 4 shows a case similar to the one in Figure 3 but where the federate time steps do not match. Here, the MPF dt is 0.015625 seconds (64 Hz) with a matching real time frame (RT). Unlike the previous case, the additional federate, Federate 1 (F1), has a dt of 0.031250 seconds (32 Hz), $dt_1 = 2 \times dt_{MPF}$. Specifically, F1 takes on a single time step for every two time steps in the MPF. In addition, $FP_1 \approx 0.013$ seconds while $FP_{MPF} \approx 0.006$ seconds.

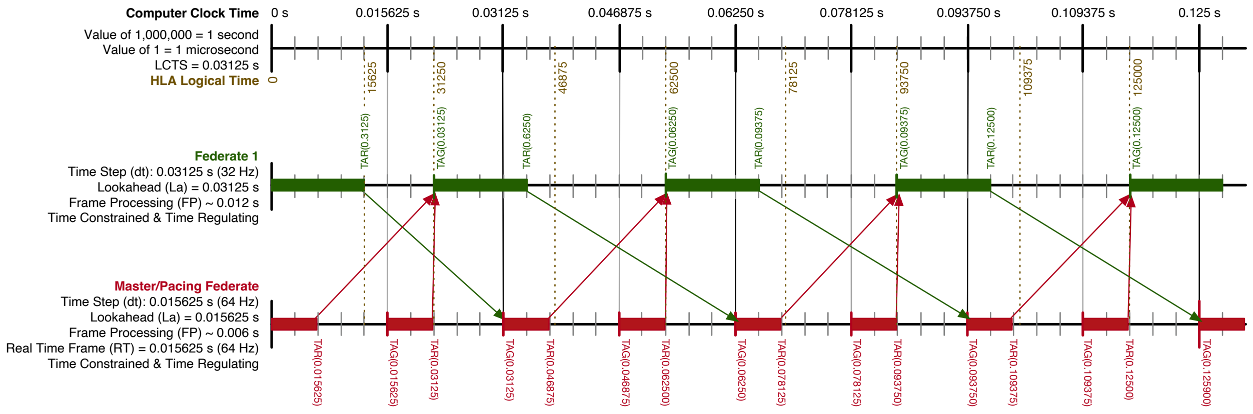


Figure 4: **HLA Time Management Simple Multi-rate Timing.**

This results in a different pattern of execution frames from those in Figure 3. In this case, F1 waits for the TAR after every other time step in MPF, even though MPF is issuing a TAR at the end of every time step. Note that F1 receives two attribute updates from MPF at the beginning of every time step while MPF only receives an attribute update from F1 at the beginning of every other time step. This is because the natural time step for F1 is twice that of MPF. Also note how the HLT values advance in relationship to the TAG points and the MPF real time frame.

4.2.5 Least Common Time Step

This is a good point to introduce another important concept called the Least Common Time Step ($LCTS$). The $LCTS$ is the least common value of all the federate time steps for the time regulating and/or time constrained federates in a federation execution. The $LCTS$ can be computed by finding the Least Common Multiple (LCM) of the set of time steps (dt_i) of the time regulating and/or time constrained federates joined to the federation execution [6, 7, 8]. The LCM function operates on integers and will require the time steps (dt_i) be converted to the integer HLT representation.

$$LCTS = LCM(HLT(dt_1), HLT(dt_2), ..., HLT(dt_N)) \tag{3}$$

The $LCTS$ is used in the computation to find the next HLT boundary available to all federates in the federation execution. For the preceding example, MPF and F1 federates share common frame boundaries at integer multiples of the $LCTS$; in this case $LCTS = 31250$ (0.03125 s).

4.2.6 Complex Multi-Rate Multi-Federate Case

The final case explored here is a more complex multi-rate multi-federate case where the federate time steps are not as simply related as above. This case starts with the MPF and F1 federates in 4.2.4 but adds an additional federate, F2, that has a time step of 0.025 seconds. The $LCTS$ for this collection is no longer 0.03125 seconds but increases to 0.125 seconds. As can be seen in Figure 5, this leads to a HLT common frame boundary every 8 time steps for the MPF, every 4 time steps for F1, and every 5 time steps for F2.

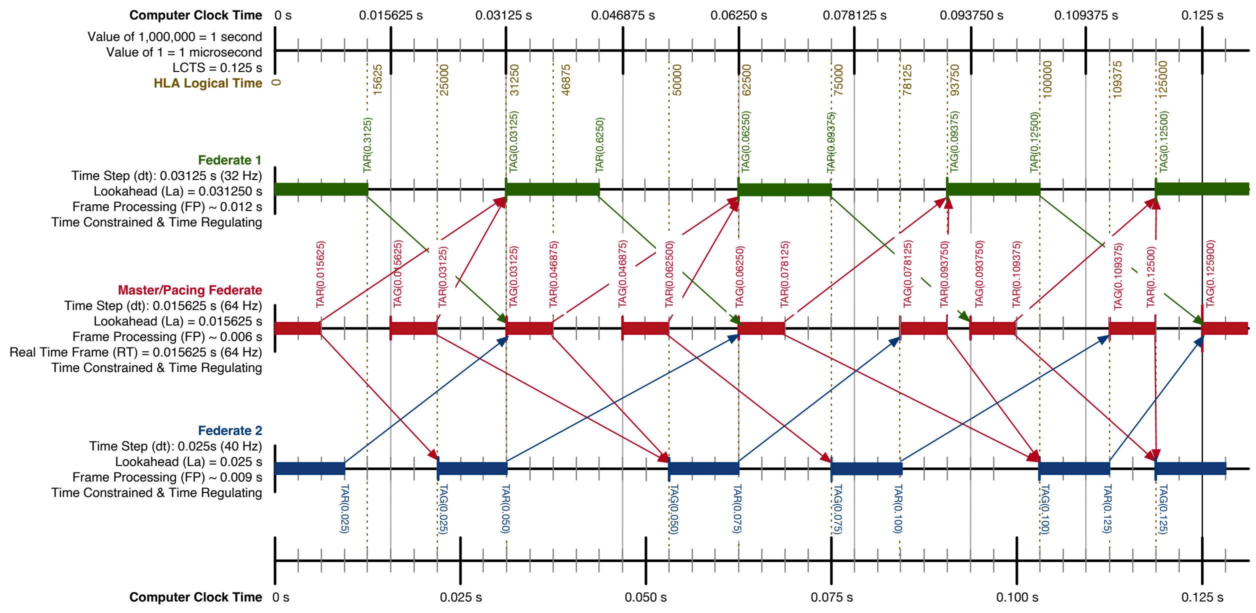


Figure 5: **HLA Time Management Complex Multi-rate Timing.**

This results in some unusual federate time step phasing based on the 0.015625 second (64 Hz) real time frame of the MPF and TAR/TAG sequences of the other federates. Figure 5 includes attributed update arrows between MFP/F1 and MFP/F2.[7] Note that the attribute update relationship between MPF and F1 maintains the same relationship as in section 4.2.4. It would be a problem if they did not. However, the time step timing changes in order to accommodate the time step of F2 and the associated TARs.

The attribute update sequence between MPF and F2 is more interesting. Note that the number of attribute updates for each time step varies from step to step for both federates. The MPF receives 1 update from F2 on some time steps and 0 updates on other time steps. Similarly, F2 received 2 updates from MPF on some time steps and only 1 update on others. This is a function of the beat between the time steps of the two federates.

While all this may appear chaotic, it is not. After the first $LCTS$ at 0.125 seconds (HLT 125000), the time step timing and the attribute updates settle into an irregular but repeatable pattern.

5 Timing Constraint Equations

How are all these timing parameters related for complex multi-federate multi-rate federation executions?

[7] Attribute updates could be included between F1/F2 but would make the diagram harder to read and do not add any additional information to this discussion.

The cases in the preceding sections can be used to develop a collection of equations that help understand the timing constraints of multi-rate multi-federate federation executions and check for timing consistency. To simply the equations it is assumed the federates are co-located so speed of light distance latencies between federates are negligible and the data transmission latency is negligible. For this discussion, the constraint relationships are organized into two categories: individual federate and federation wide constraints.

5.1 Individual Federate Constraints

The first category of constraint equations deal with the individual time regulating and/or time constrained federates. These equations deal with the individual federate Lookahead (La_i) and time step (dt_i) values in association with the federation wide Least Common Time Step ($LCTS$) (see Equation 3).

The first two equations deal with the federate time step (dt_i) and Lookahead value (La_i). In HLA, the lookahead value is a nonnegative HLT value that establishes a lower value on the timestamps that can be sent in timestamp order (TSO) messages by joined time regulating federates. The first relationship is the federate time step (dt_i) must be strictly greater than zero so that federate time will advance.

$$dt_i > 0 \tag{4}$$

The second relationship is between La_i and the federate time step (dt_i); dt_i must be greater than or equal to La_i.[8]

$$dt_i \geq La_i \tag{5}$$

A time step (dt_i) less then the Lookahead time (La_i) will result in an invalid requested HLT because the minimum allowable requested time will be the granted HLT plus the La_i.

The next two equations deal with $LCTS$ and the federate time step (dt_i). First, $LCTS$ must be greater than or equal to dt_i.

$$LCTS \geq dt_i \tag{6}$$

The second is that the $LCTS$ must be an integer multiple of dt_i.

$$LCTS \ \% \ dt_i = 0 \tag{7}$$

These relationships are already a consequence of the way $LCTS$ is computed but are stated here explicitly as additional checks on timing computations and constraints.

5.2 Federation Wide Constraints

Federation wide timing constraints result from the analysis of the collection of all participating time regulating and/or time constrained federated joined to the federation execution. These deal with the relationships between the $LCTS$ and the Real Time frame (RT). First, the $LCTS$ must be greater than or equal to RT.

$$LCTS \geq RT \tag{8}$$

In addition, $LCTS$ needs to be an integer multiple of RT.

$$LCTS \ \% \ RT = 0 \tag{9}$$

Both equations 8 and 9 must hold and result in the following Boolean equation:

$$(LCTS \geq RT) \wedge (LCTS \ \% \ RT = 0) \tag{10}$$

These equations are important in defining the $LCTS$ to RT relationship that will ensure that a common federate boundary and any potential SpaceFOM mode transitions fall on a real time frame boundary. This is necessary for the Master federate and the Pacing federate to coordinate federation execution with a real time clock.

[8]While HLA does support zero lookahead values, zero lookahead federates use the Time Advance Request Available API and will not be addressed in this discussion.

Finally, collecting the individual federate time constraint relationships defined in Equations 4, 5, 6, and 7 into a Boolean logical operator across the set of all joined time regulating and/or time constrained federates results in the following Boolean equation:

$$\bigwedge_{i=1}^{N} \big((dt_i > 0) \wedge (dt_i \geq La_i) \wedge (LCTS \geq dt_i) \wedge (LCTS \ \% \ dt_i = 0)\big) \tag{11}$$

This completes the set of timing constraint equations. Equations 10 and 11 can be used to preform checks on timing constraints for HLA-based distributed simulations.[9] This would normally be done during the startup and initialization phase of a federation execution.

6 Conclusions

Time synchronized execution of HLA-based distributed simulations can be a complex topic. This paper addresses aspects of that topic with a discussion on the challenges of multi-rate multi-federate time synchronization and also presents a collection of time constraint equations that can be used to check the timing consistency for individual federates and across a collection of federates in a federation execution.

A number of increasingly complex example cases are diagramed and discussed in Section 4. These are useful in understanding the relationship between various fundamental timing parameters associated with time synchronized federation executions using the HLA Time Management services. They also illustrate the nonintuitive nature of time synchronization, time advance, real time execution, and timestamp ordered data delivery. The simple matched rate multi-federate and simple multi-rate multi-federate designs are preferable to the complex multi-rate multi-federate design.

The Challenges discussion in Section 4 is followed by a discussion of time constraint equations in Section 5. The equations address time constraint relationships with individual federates and across the set of time managed federates in a federation execution.

It is important that HLA/SpaceFOM federate and federation developers understand the time synchronization fundamentals of multi-rate, multi-federate, time managed, and real time federation executions along with the associated time constraints involved. Developers should consider generating timing diagrams like those in Section 4 when designing a federation execution and checking the time constraint equations shown in Section 5 with the federate startup and initialization routines to ensure timing consistency.

7 Future Work

Much of the content of this paper has been developed by engineers in the Software, Robotics, and Simulation Division (ER) at NASA's Johnson Space Center. ER develops and maintains a number of NASA open source simulation software packages including a simulation development environment called Trick [9] and an HLA/SpaceFOM middleware package called TrickHLA [10]. The time constraint checks described in Section 5 are being incorporated into the TrickHLA initialization code to ensure time synchronization consistency. The timing diagram techniques will be used to help design a distributed simulation architecture for NASA's Artemis program.

In future work, the time constraint equations could be updated to account for latencies arising from the physical distances between federates and the data transmission latencies coming from the data sizes and network speeds. Real time constraints could be derived from the federate frame processing times and federate time step sizes while using a timing diagram to determine the worst case number of frames processed within a real time frame.

[9]The federate timing constraint equations do allow for a zero lookahead value, but it will not be addressed in this discussion.

Acronyms

API	Application Programming Interface
dt	Federate Time Step
CCT	Computer Clock Time
ER	The Software, Robotics, and Simulation Division
F1	Federate #1
F2	Federate #2
FOM	Federation Object Model
FP	Frame Processing time
FST	Federation Scenario Time
HITL	Human-In-The-Loop
HLA	High Level Architecture
HLT	HLA Logical Time
HWITL	Hardware-In-The-Loop
Hz	Cycles per second
IEEE	Institute for Electrical and Electronics Engineers
JSC	Johnson Space Center
La	Lookahead time
LCTS	Least Common Time Step
M&S	Modeling and Simulation
MPF	Master/Pacing Federate
NASA	National Aeronautics and Space Administration
NExSyS	NASA Exploration Systems Simulations team
NRT	Non-Real-Time
PT	Physical Time
RT	Real-Time frame in Pacing Federate
RTI	Runtime Infrastructure
SET	Simulation Elapsed Time
SISO	Simulation Interoperability Standards Organization
SST	Simulation Scenario Time
SpaceFOM	Space Reference Federation Object Model
TSO	Time Stamp Ordered
TAG	Time Advance Grant
TAR	Time Advance Request

References

[1] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.* Technical Report IEEE-1516-2010, The Institute of Electrical and Electronics Engineers, 2 Park Avenue, New York, NY 10016-5997, August 2010.

[2] E. Crues, D. Dexter, A. Falcone, A. Garro, M. Madden, B. Möller, and D. Ronnfeldt. *Standard for Space Reference Federation Object Model (SpaceFOM).* Number SISO-STD-018-2020. Simulation Interoperability Standards Organization (SISO), P.O. Box 781238, Orlando, FL 32878-1238, USA, October 2019.

[3] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification.* Technical Report IEEE-1516.1-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.

[4] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification.* Technical Report IEEE-1516.2-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.

[5] Edwin Z. Crues, Dan Dexter, Alberto Falcone, Alfredo Garro, and Björn Möller. Spacefom - a robust standard for enabling a-priori interoperability of hla-based space systems simulations. *Journal of Simulation*, 16(6):624–644, 2022.

[6] Eric W. Weisstein. Least common multiple. MathWorld: `https://mathworld.wolfram.com/LeastCommonMultiple.html`, December 2024.

[7] Eric W. Weisstein. Greatest common divisor. MathWorld: `https://mathworld.wolfram.com/GreatestCommonDivisor.html`, December 2024.

[8] Eric W. Weisstein. Prime factorization. MathWorld: `https://mathworld.wolfram.com/PrimeFactorization.html`, December 2024.

[9] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7). Trick simulation environment: Documentation. Trick Wiki Site: `https://nasa.github.io/trick/documentation/Documentation-Home`, September 2023.

[10] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7). TrickHLA. TrickHLA GitHub Site: `https://github.com/nasa/trickhla`, January 2025.

Author Biographies

Edwin Z. {Zack} Crues: *received B.S., M.S., and Ph.D. degrees in Aerospace Engineering from the University of Texas, Austin in 1983, 1985, and 1989 respectively. Zack has over 35 years of professional experience in developing spacecraft simulation and simulation technologies internationally. Zack currently works at NASA Johnson Space Center in Houston, Texas as a Space Systems Simulation Architect in the Simulation and Graphics Branch where he leads the development of simulation technologies and the application of those technologies in the simulation of NASA's current and proposed crewed spacecraft.*

Daniel E. Dexter: *is an engineer in the Spacecraft Software Engineering Branch in the Software, Robotics and Simulation Division of the Engineering Directorate at NASA's Johnson Space Center in Houston, Texas. Daniel has over 30 years of software and simulation development experience including the areas of digital signal processing, image processing, neural networks, distributed supercomputing, embedded flight software, and distributed simulations. Daniel is the principal developer of the TrickHLA software package, a NASA developed IEEE 1516 simulation interoperability middleware for the NASA Trick simulation environment M&S tool.*