Time Management in the DoD High Level Architecture

Richard M. Fujimoto
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Richard M. Weatherly
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102-3481

Abstract

Recently, a considerable amount of effort in the U.S. Department of Defense has been devoted to defining the High Level Architecture (HLA) for distributed simulations. This paper describes the time management component of the HLA that defines the means by which individual simulations (called federates) advance through time. Time management includes synchronization mechanisms to ensure event ordering when this is needed. The principal challenge of the time management structure is to support interoperability among federates using different local time management mechanisms such as that used in DIS, conservative and optimistic mechanisms developed in the parallel simulation community, and real-time hardware-in-the-loop simulations.

1. Introduction

The Defense Modeling and Simulation Office (DMSO), through its High Level Architecture (HLA) initiative, is addressing the continuing need for interoperability between new and existing simulations within the U. S. Department of Defense. The HLA seeks to generalize and build upon the results of the Distributed Interactive Simulation (DIS) world and related efforts such as the Aggregate Level Simulation Protocol (ALSP) [Wils94]. The HLA activity began in March 1995 with the goal of recommending an architecture to the Executive Council for Modeling and Simulation (EXCIMS) before the end of calendar year 1996. The EXCIMS in turn, after appropriate review, will recommend the architecture to the Under Secretary of Defense (Acquisition and Technology) for approval and standardization. Prototype demonstrations of the use of the architecture are scheduled to be completed in the summer of 1996. Information about the HLA concept and the DMSO Master Plan is available at http://www.dmso.mil.

The HLA consist of three parts: 1) rules governing certain characteristics of HLA-compliant simulations, 2) an object modeling scheme that describes the information of common interest to a group (called a federation) of cooperating simulations (federates), and 3) the Run-Time Infrastructure (RTI) that provides the software environment needed by the federates to exchange information in a coordinated fashion. The RTI is a special purpose distributed operating system that provides a variety of services, described below. The specification of these services is evolving through experimentation and can be found on the web server mentioned above. In this paper we describe the time management services. The principal challenge is to bring together, in a general and extensible way, the time management mechanisms used by several disparate communities including DIS, ALSP, and test and evaluation. Below, we briefly review key concepts in DIS and ALSP before describing the HLA.

"The primary mission of DIS is to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual `worlds' for the simulation of highly interactive activities" [DIS94]. A DIS exercise can be viewed as a collection of autonomous simulations each maintaining a virtual environment representing the portions of the battlefield relevant to the entities it is modeling. An exercise may include (1) human-in-the-loop elements such as tank or flight simulators, (2) computation only elements such as wargame simulations, and (3) live elements such as instrumented tanks. Time advances are paced by a real-time clock. State changes, e.g., firing a weapon, are broadcast as they occur. Each element determines what information is relevant to the entities it models and discards the rest. Messages are typically processed in receive order (not time stamp order) to reduce communication latency, sometimes leading to anomalies. Some temporal errors are acceptable because they will not be noticed due to limitations in human perception. Unreliable communication services are often used, again to reduce latency. See [DIS94] for an introduction to DIS, and [Fuji95] for a discussion contrasting DIS and parallel simulation research.

ALSP was designed to extend the DIS concept, and focused largely on combining separately developed wargame simulations into federations. Wargame simulations are often referred to as constructive or aggregated simulations because they model battlefield components at a higher, more aggregated level of abstraction, e.g., battalions or divisions rather than individual aircraft or tanks. A key distinction between ALSP and the training simulations used in DIS is ALSP federations require strict adherence to causality, i.e., simulation events must be processed in time stamp order. ALSP currently uses the Chandy/Misra/Bryant null message protocol to accomplish this [Chan79].

2. Overview of the HLA

Real-world entities are modeled in the HLA by objects. Each object contains an identifier, state, and a behavior description that specifies how the object reacts to state changes. The relationship of objects to one another is specified through (1) attributes that indicate those state variables and parameters of an object that are accessible to other objects, (2) association between objects (e.g., one object is part of another object), and (3) interactions between objects that indicate the influence of one object's state on the state of another object. A *federation object model (FOM)* specifies the common object model used by all federates.

Each object attribute has an owner that is responsible for updating the value of its attributes (e.g., position information). At any instant, there can be at most one owner of an attribute, however, ownership of the attribute may pass from one federate to another during an execution. Other federates *subscribe* to receive updates to attributes as they are produced by the owner.

The runtime component defines a set of services invoked by federates or by the Run-Time Infrastructure (RTI) during a federation execution. HLA runtime services fall into the following categories:

- *Federation management.* This includes services to create and delete federation executions, to allow federates to join or resign from an execution, and to pause, checkpoint, and resume an execution.
- *Declaration management.* These services provide the means for federates to establish their intent to publish object attributes and interactions, and to subscribe to updates and interactions produced by other federates.
- *Object management.* These services allow federates to create and delete object instances, and to produce and receive attribute updates and interactions.
- *Ownership management.* These services enable the transfer of ownership of object attributes during the federation execution.
- *Time management.* These services coordinate the advancement of logical time, and its relationship to wallclock time during the federation execution.

The remainder of this document is concerned with the time management services. See [DMSO96] for additional information concerning time management.

3. Time Management Interoperability

The RTI provides a base into which separately developed simulations can be "plugged in" to form large distributed simulations. A central goal of the high level architecture time management (HLA-TM) structure is to support interoperability among federates utilizing different internal time management mechanisms. Specifically, a single federation execution may include:

1. federates with different event ordering requirements, e.g., DIS and ALSP federates.
2. federates using different time flow mechanisms, e.g., timestepped and event driven mechanisms.
3. real-time (or scaled real-time) and as-fast-as-possible simulations; it is assumed that individual federates executing in conjunction with the RTI perform at least as fast as scaled wallclock time in federations requiring (scaled) real-time execution.
4. federates executing on parallel/distributed platforms using conservative or optimistic synchronization.
5. federates using a mixture of event ordering and transportation services, e.g., a DIS-like federate may use time stamp ordering and reliable message delivery for certain types of events, and receive-ordered, best-effort delivery for others. This facilitates gradual, evolutionary exploitation of previously unused HLA-TM services.

Time management transparency is important to achieve interoperability. This means the time management mechanism used within each federate is *not* visible to other federates.

4. Events, Messages, and Time

An execution can be viewed as a collection of federates, each performing a sequence of computations. Some of these computations are referred to as *events*, and some of these events are relevant to other federates. The RTI notifies other federates that have indicated an interest in an event by sending a *message* for each event notifying the federate the event has occurred. The time stamp of the message refers to the time stamp of the corresponding event. *Events* and *messages* are *not* synonymous; a single event typically produces many messages to notify other federates of the event. Lookahead constraints are also placed on events that are to be delivered in time stamp order, as discussed later. Federates need not schedule events in time stamp order. There are four types of events in the HLA: creation of a new object, deletion of an object, a state update, or an interaction.

Time in the system being modeled is represented in the HLA by a global *federation time axis*. The federation time axis is defined as a totally ordered sequence of values where each value represents an instant of time in the physical system being modeled, and for any two points T_1 and T_2 on the federation time axis, if $T_1 < T_2$, then T_1 represents an instant of physical time that occurs before the instant represented by T_2.

Two separate clocks are defined within each federate: *scaled wallclock time* is used to synchronize the execution with humans and live entities, and *logical time* is used to ensure that messages are delivered in a proper temporal sequence. Both represent points on the federation time axis. Scaled wallclock time is defined as offset + [rate *(wallclock time - time of last exercise start or restart)]. *Wallclock time* is defined as a federate's measurement of true global time and is typically output from a hardware clock. If *rate* is k, scaled wallclock time advances k time faster than wallclock time. Non-real-time (aka as-fast-as-possible) executions set the rate factor to infinity.

Logical time is synonymous with "simulated time" in the parallel simulation literature, and is only relevant to federates that require that messages are not delivered to the federate "in its past," i.e., with time stamp smaller than the federate's current time. Federates must explicitly request advances in logical time. The requirement that messages are not delivered "in the past" only applies to messages that are not designated to be delivered in time stamp order. Federates not requiring this constraint (e.g., DIS federates) request a time advance to "infinity" at the beginning of the execution.

Federate time denotes the "current time" of the federate, and these two terms are used synonymously. Federate time is defined as scaled wallclock time or logical time of the federate, whichever is smaller. At any instant of an execution different federates will, in general, have different federate times.

5. HLA-TM Services

Time management is concerned with the mechanisms for controlling the advancement of time during the execution of a federation. Time advancement mechanisms must be coordinated with other mechanisms responsible for delivering information, e.g., to ensure messages are not delivered in a federate's past. Thus, the time management services must encompass two aspects of federation execution:

- *Transportation services:* Different categories of service are specified that provide different reliability, message ordering, and cost (latency and network bandwidth consumption) characteristics.
- *Time advancement services:* Different primitives are provided for federates to request advances in logical time. These primitives provide the means for federates to coordinate their time advances with the time stamp of incoming information, if this is necessary. The time advance mechanism in the RTI must accommodate both scaled real-time, and as-fast-as-possible executions.

5.1 Transportation Services

The different categories of transportation service are distinguished according to (1) reliability of message delivery, and (2) message ordering. With respect to reliability, *reliable message delivery* means the RTI utilizes mechanisms (e.g., retransmission) to increase the probability that the message is eventually delivered to the destination federate. This improved reliability normally comes at the cost of increased latency. On the other hand, the *best effort message delivery* service attempts to minimize latency, but with the cost of lower probability of delivery.

Message ordering characteristics specify the order and time at which messages may be delivered to federates and are central to the HLA time management services. A variety of services are provided to support interoperability among federates with diverse requirements. Five ordering mechanisms are currently specified in the HLA: receive, priority, causal, causal and totally ordered, and time stamp ordered. These provide, in turn, increased functionality but at increased cost.

The ordering mechanisms currently defined are:

- *Receive Order.* Messages are passed to the federate in the order that they were received. Logically, incoming messages are placed at the end of a first-in-first-out (FIFO) queue, and are passed to the federate by removing them from the front of this queue. This is the most straightforward, lowest latency ordering mechanism.
- *Priority Order.* Incoming messages are placed in a priority queue, with the message time stamp used to specify its priority. Messages are passed to the federate lowest time stamp first. This service does not prevent a message from being delivered in a federate's "past" (time stamp less than the federate's current time), but it is less costly in terms of latency and synchronization overhead than the time stamp ordered delivery mechanism. Priority order with best effort delivery may be used for federates where sequences of messages require ordering, but the increased latency associated with either reliable delivery or guaranteed order cannot be tolerated. For example, speech packets may utilize this service.
- *Causal order.* This service guarantees that if an event E "causally precedes" another event F, then any federate receiving messages for both events will have the message for E delivered to it before the message for F. For example, E and F might indicate firing a weapon, and the target being destroyed, respectively; if causal ordering is used, a federate observing both events will be notified of

the fire event before it is notified of the destroyed event.

The "causally precedes" relationship is identical to Lamport's "happens before" relationship [Lamp78]. This relationship is defined between a pair of actions A_1 and A_2, where an action is an event, and RTI message send, or an RTI message receive. This relationship (denoted $\rightarrow$) is defined as follows: (i) if A_1 and A_2 occur in the same federate/RTI, and A_1 precedes A_2 in that federate/RTI, then $A_1 \rightarrow A_2$, (ii) if A_1 is a message send action and A_2 is a receive action for the same message, then $A_1 \rightarrow A_2$, and (iii) if $A_1 \rightarrow A_2$ and $A_2 \rightarrow A_3$, then $A_1 \rightarrow A_3$ (transitivity).

- *Causal and totally ordered.* In the causally ordered service defined above, messages corresponding to events that are *not* causally related (referred to as concurrent events) may be delivered to federates in any order. The causal and totally ordered service extends causal ordering to guarantee that for any pair of concurrent events, messages for these events will be delivered to all federates receiving both messages *in the same order*, thereby defining a total ordering of events. This service is commonly referred to as CATOCS (causally and totally ordered communications support) in the literature (e.g., see [Birm91]).

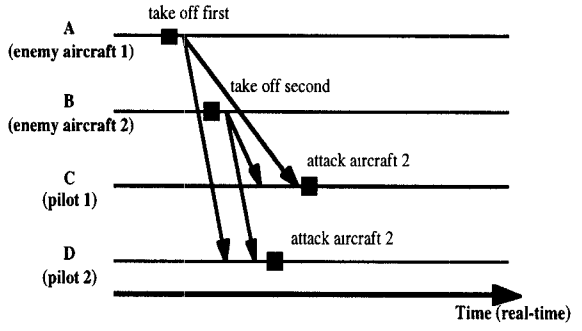

Figure 1. Scenario demonstrating causal and total ordering.

Figure 1 illustrates where CATOCS may be useful. Two federates (A and B) modeling enemy aircraft are taking off from an air field. Two pilots are assigned to intercept, with pilot 1 (federate C) given orders to attack the first enemy aircraft to take off, and pilot 2 (federate D) assigned to attack the second. Assume the "take-off" events are concurrent, e.g., the aircraft are taking off from different runways. Without total ordering, messages for the take-off events may arrive at the two federates in different orders. Figure 1 shows a scenario where pilot 1 incorrectly believes aircraft 2 took off first, while pilot 2 correctly believes aircraft 1 took off first. The end result is both pilots attack aircraft 2! CATOCS would circumvent this anomaly by ensuring that both pilots see the same ordering of events. Both pilots may perceive an incorrect order (e.g., both might believe aircraft 2 took off first if the messages from Federate A are delayed), but the result of this error is likely to be less severe than having both pilots attack the same aircraft.

- *time stamp order (TSO).* Messages utilizing this service will be delivered to federates in time stamp order. Further, the RTI also ensures that no message is delivered to a federate "in its past," i.e., no TSO message is delivered that contains a time stamp less than the federate's current time. A conservative synchronization protocol is used to implement this service. All federates receiving messages for a common set of events will receive those messages in the same order, i.e., a total ordering of events is provided. The RTI provides a consistent tie breaking mechanism so messages containing identical time stamps will be delivered to different federates in the same order. Further, the tie-breaking mechanism is deterministic, meaning repeated executions of the federation will yield the same relative ordering of these events if the same initial conditions and inputs are used, and all messages are transmitted using time stamp ordering.

The relationship between causal and time stamp order is described in greater detail in [Fuji96]. The above orderings yield, in turn, successively stronger guarantees concerning message ordering, at the cost of increased latency, communication bandwidth, and in the case of time stamp ordering, constraints on scheduling events. Each federate may intermix different message ordering services for different types of information within a single federation execution. For example, periodic position updates may utilize a best effort, receive order category of service. These messages may be intermixed with messages for ordnance detonation events utilizing reliable, time stamp ordered delivery.

5.2 Object Management Services

The object management services include primitives to schedule and retract events, and primitives to receive messages. For example, **Update Attribute Values** and **Send Interaction** schedule events. The sender assigns a time stamp to the event to indicate when it is to occur, and specifies the category of transportation service (reliability and message ordering) that is to be used. The RTI delivers messages to the federate by invoking services that must be provided by the federate. Specifically, the RTI invokes the federate's **Reflect**

Attribute Values and **Receive Interaction** services to deliver messages denoting state changes and interactions.

Event retraction refers to the ability of a federate to retract or unscheduled a previously scheduled event. This is a common discrete-event simulation primitive often used to model interrupts and other preemptive behaviors. As discussed later, event retraction is also utilized by optimistic federates to implement anti-messages. The **Update Attribute Values** and **Send Interaction** services return an event handle that is used to specify the event that is to be retracted.

If the RTI at the destination federate receives a retraction request for an event that is not buffered in the RTI (e.g., because the corresponding message has already been forwarded to the federate), the retraction request is forwarded to the federate.

5.3 Lookahead

The time stamp order service requires specification of lookahead. Here, lookahead is defined as the minimum distance into the future that a TSO event will be scheduled. A lookahead value is associated with each federate. If the federate's lookahead is L, all TSO events must have time stamp of at least the federate's current time plus L.

Lookahead can change dynamically during the execution. However, lookahead cannot instantaneously be reduced. At any instant, a lookahead of L indicates to the RTI that the federate will not generate any event (using time stamp ordering) with time stamp less than C+L, where C is the federate's current time. If the lookahead is reduced by K units of time, the federate must advance K units before this changed lookahead can take effect, so no events with time stamp less than C+L are produced.

A federate's lookahead must be strictly greater than zero. This is necessary because if the RTI has advanced the federate's logical time to T, then (in most cases) it guarantees that all TSO messages with time stamp less than *or equal to* T have been delivered to the federate. This would be impossible to guarantee if lookahead were zero because a federate A at logical time T could schedule an event with time stamp T that is received by another federate B, which in turn schedules a second event also with time stamp T that is received by A, violating the guarantee that the RTI had delivered all TSO messages with time stamp T or less. The RTI will always use a small, nominal value for lookahead to circumvent situations such as this.

Lookahead restrictions must be applied to retracting previously scheduled events utilizing the time stamp ordered message delivery service if it is important that messages for retracted events never be passed on to other federates. Specifically, if the current time of a federate is T, and its lookahead is L, then the federate can only retract events containing time stamps greater than T+L. Messages for events containing a smaller time stamp may have already been passed to other federates.

It is possible that a message for a retracted event may be lost in the network, especially if best effort delivery is used. In this case, if the retract request was successfully delivered, the message for the retracted event will never appear, yet the retract request would still be passed to the receiving federate. Federates must be designed to allow for situations such as this. The RTI does guarantee that if a retraction request is forwarded to a federate, the retracted message will not be later delivered to the federate.

5.4 Time Advance Services

The time advance primitives serve several purposes. First, it provides a protocol for the federate and RTI to jointly control the advancement of logical time. The RTI can only advance the federate's logical time to T when it can guarantee that all TSO messages with time stamp less than or equal to T have been delivered to the federate. At the same time, the federate must delay processing any local event until logical time has advanced to the time of that event, or else it is possible it will receive a TSO message in its past.

The time management primitives also control the delivery of messages to the federate. TSO messages will not be delivered until the receiving federate has requested a time advance up to at least the time stamp of the message. In addition, the time management primitives provide information to the RTI that is used to synchronize the execution, as discussed below.

Two services for advancing logical time are defined: **Time Advance Request** and **Next Event Request**. Here, we assume these are invoked to request that all eligible messages are delivered to the federate. A provision is also provided to only deliver one message at a time. **Time Advance Request** is intended to be used by time-stepped federates, and **Next Event Request** by event-driven federates. Each invocation of either primitive eventually results in the RTI calling **Time Advance Grant** to indicate that logical time has been advanced, and all TSO messages with time stamp less than or equal to the grant have been delivered.

Time Advance Request with parameter t requests an advance of the federate's logical time to t. When used in a time-stepped simulation, t will usually

indicate the time of the next time step. Invocation of this service implies that the following messages are eligible for delivery to the federate: (i) all incoming receive ordered messages, and (ii) all messages using other ordering services with time stamp less than or equal to t. The federate may simply note the occurrence of these events for later processing, or immediately simulate actions resulting from the occurrence of the events. When the RTI can guarantee that it has passed all TSO messages to the federate with time stamp less than or equal to t, logical time is advanced to t, and the RTI calls the federate's **Time Advance Grant** primitive. At this point, a time-stepped federate may proceed to simulate the next time step.

Next Event Request with time parameter t requests an advance of logical time to t, *or the time stamp of the next TSO message from the RTI*, whichever is smaller. When used in an event driven federate, t will usually indicate the time stamp of the next local event within the federate. After the primitive is invoked, the federate will either (i) deliver the next TSO message (and all other TSO messages containing exactly the same time stamp) if that message has a time stamp of t or less, and advance logical time to the time of that message, or (ii) not deliver any TSO messages and advance logical time to t. In either case, a **Time Advance Grant** is issued to indicate completion of the request. Other non-TSO messages may also be delivered as a result of invoking this primitive. If no TSO messages are delivered as a result of this request, this indicates to the federate that it may process its local event with time stamp t because no externally generated TSO messages with time stamp less than or equal to t are forthcoming.

In an as-fast-as-possible execution if a federate invokes **Time Advance Request** with parameter t, it unconditionally guarantees 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. By invoking **Next Event Request** with parameter t, the federate is making a conditional guarantee that if it does not receive any additional TSO messages in the future with time stamp less than t, the federate will not later generate any TSO messages with time stamp less than t plus the federate's lookahead. This information is not unlike that used in the framework described in [Jha94].

It is noteworthy that as defined above, **Time Advance Request**, **Next Event Request**, and **Time Advance Grant** only pertain to the advancement of logical time. Wallclock time advances independent to the federate's actions (of course!). The **Time Advance Grant** call (or any action that advances logical time) will only be made after the RTI can guarantee no future messages will arrive with time stamp less than or equal to the federate's new logical time. The time required to make this advance depends on the performance of the synchronization protocol.

6. Synchronization Protocol

A conservative synchronization protocol implements the time stamp order message delivery service, and is used to advance logical time. The principal task of the protocol is to determine a value called LBTS for each federate, defined as a lower bound on the time stamp of future TSO messages that it will receive from other federates. Any TSO message with time stamp less than LBTS is eligible for delivery to the federate. The logical time of the federate cannot be advanced beyond LBTS. The specific synchronization protocol that is used is not visible to the federate to facilitate later inclusion of new protocols. Simulations that are dependent on the particular protocol that is used are considered non-HLA-compliant.

It is instructive to outline one *possible* implementation of the synchronization protocol. The following describes an implementation that is a variation of the well-known Chandy/Misra/Bryant "null message" algorithm. It is assumed that communications are reliable, and messages sent from one processor to another are delivered in the order that they were sent. This is important to prevent null messages from "passing" messages with smaller time stamps.

Federates are not constrained to send messages in time stamp order. Thus, the time stamp of incoming messages do not provide useful information that can be used by the synchronization protocol. Instead, null messages must be relied upon to carry all synchronization information.

LBTS for a federate F is computed as the minimum ($current\_time_i$ + $lookahead_i$) computed over all other federates that send TSO messages to F. In an as-fast-as-possible execution, "current time" is identical to logical time, so LBTS can be maintained if each federate sends a null message each time it advances in logical time. However, in a real-time execution, current time advances with scaled wallclock time if logical time remains larger than scaled wallclock time. Further, DIS federates may not utilize logical time. Thus, it is not feasible to send a null message with each current time advance. This problem can be solved by observing that each federate can determine a lower bound on scaled wallclock time in each other federate provided the difference in wallclock values is bounded.

The synchronization protocol consists of two components: a sender and a receiver component. They operate as follows:

1. After each (local) logical time advance, the sender component transmits a null message to each destination to which it sends TSO messages with time stamp equal to the federate's current logical time, plus a lookahead value. This time stamp indicates a lower bound on the time stamp of any TSO message the federate will send in the future.
2. The RTI for each federate maintains a local bound for messages received from each source i called LB[i]. Upon receiving a null message for source *i*, the receiving RTI sets LB[i] equal to the time stamp of the null message. If this causes LBTS to change (increase; LBTS can never decrease), LBTS is also modified. This may, in turn, cause an increase in logical time, resulting in additional null messages.
3. In a real-time federation execution, each RTI also determines a lower bound on the wallclock time of neighboring federates (federates that send it messages) provided skew between the wallclock time of different simulations is bounded. This, along with lookahead information, enables the RTI to determine a lower bound on the time stamp of future TSO messages sent from other federates (even DIS federates) and can be used to set LB[i]. In general, LB[i] may be defined as the minimum of (1) the predicted real-time clock of simulation *i* plus its lookahead, and (2) the time stamp of the last null message received from simulation *i*.

7. Optimistic Time Management Services

The HLA-TM services described next are intended to enable optimistic federates to utilize HLA-TM services while still enjoying the advantages afforded by optimistic execution. These services *do not require all federates to support a rollback and recovery capability.* Indeed, it is envisioned that federations may include *both* optimistic and conservative federates within a single execution. Conservative federates not needing or desiring to utilize optimistic processing techniques may completely ignore the optimistic time management services with no ill effects.

An important goal of the optimistic time management services is to enable optimistic messages to be delivered to other optimistic federates (but not conservative federates). Thus, simple solutions such as requiring that the optimistic simulation *only* send messages that it can guarantee will not be later canceled are undesirable, because they do not fully exploit the potential offered by optimistic execution.

Several modest additions to the above services are used to support optimistic execution, as described below. These are collectively referred to as optimistic time management services. The discussion that follows only pertains to TSO messages.

1. Simulations may receive TSO messages *before* the RTI can guarantee that no smaller time stamped messages will be later received, i.e., before the RTI can guarantee time stamp ordered delivery.
2. An RTI primitive is provided for the federate to indicate to the RTI its logical time value, as discussed below.
3. The LBTS value is made available to the optimistic federate.

To illustrate how these services can be used by an optimistic simulation, the following outlines how a Time Warp [Jeff85] based simulation (TW) could be included in an HLA federation:

- The TW simulation uses the optimistic event facility to receive, and optimistically process events.
- Optimistically generated messages are transmitted through the RTI to other federates, the same as ordinary, non-optimistic messages. The RTI does not distinguish between optimistic and conservative events.
- The event retraction primitive provided by the RTI is used to cancel optimistic messages that later prove to be incorrect.
- If the canceled event has not been delivered by the RTI to the receiving simulation, annihilation happens within the RTI. If the message has already been delivered to the receiving simulation, the retraction request is forwarded to the simulation which must perform the cancellation itself, typically by performing a rollback in the receiving federate, possibly generating additional cancellation (retraction) requests.
- The RTI's conservative synchronization mechanism is used to prevent conservative federates from receiving optimistic messages. Specifically, the logical time of an optimistic federate is set equal to the GVT of the federate. Any event with time stamp less than GVT is guaranteed not to be prone to future rollbacks, and since events must be generated at least L time units into the future, where L is the lookahead, any event with time stamp less than GVT+L is guaranteed not to be subject to any future rollback. In effect, the GVT acts as the "local clock" for the TW federate from the perspective of the RTI. Lookahead is only used in the optimistic federate to enhance the performance of the GVT computation.
- Within the Time Warp simulation, GVT is computed as the minimum of the local GVT, and LBTS, provided by the RTI. This is because LBTS indicates a lower bound on the time stamp of any future TSO message, so it therefore provides

a lower bound on the time stamp of any future rollback caused by receiving a message (or anti-message) in the federate's past.

A key property of this approach is it enables Time Warp federates to "plug into" the RTI, without any other federate (even optimistic ones) realizing there is an optimistic federate in the execution. The RTI allows for optimistic exchange of messages among optimistic federates, and at the same time, guarantees that optimistic messages are not released to conservative federates, all transparent to the federates participating in the execution. Further, no special GVT messages must be exchanged between optimistic federates, as the RTI automatically provides the information necessary for each optimistic federate to compute GVT locally. Finally, another attractive feature of this approach is it requires only modest modification of the "conservative" time management services already specified in the RTI.

One limitation of the above approach is it assumes a receive time stamp based definition of GVT. Some memory management protocols (e.g., Cancelback, message sendback) require a somewhat different definition of GVT. Some modifications to the above mechanism are required to support this alternate definition of GVT for optimistic federates using such techniques.

8. Conclusions

Thus far, research in the DIS and parallel simulation communities have proceeded largely independent of one another. Indeed, because of their different goals and requirements, there has been little reason for techniques in one domain to find application in the other. However, as DIS expands to encompass simulations requiring causality and event ordering, new opportunities arise for research in the parallel simulation community to have a large impact in future distributed simulation systems. The DoD High Level Architecture effort provides a framework into which research from the parallel simulation community can readily impact real-world systems.

9. Acknowledgments

The time management approach used in the HLA is the result of the collective efforts of many individuals, including Judith Dahmann, the technical lead of the HLA effort, and the members of the HLA time management working group. Individuals contributing to this design include David Bruce, Chris Carothers, Danny Cutts, Charles Duncan, Jerry Dungee, Jean Graffagnini, Richard Henderson, Jack Kramer, Michael Langen, Margaret Loper, Larry Mellon, Henry Ng, Ernie Page, Kiran Panesar, Les Parish, Dana Patterson, E. L. Perry, Jerry Reaper, Paul Reynolds Jr., Sudhir Srinivasan, Jeff Steinman, Bill Stevens, and Darrin West. Sudhir Srinivasan suggested inclusion of causal ordering in the HLA. Richard Fujimoto's work as chair of the HLA Time Management group was funded by the Defense Modeling and Simulation Office (DMSO).

10. References

[Birm91] K. Birman, A. Schiper and P. Stephenson, Lightweight Causal and Atomic Group Multicast, *ACM Transactions on Computer Systems,* 9(3): 272-314, August 1991.

[Chan79] K. M. Chandy and J. Misra, "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs," IEEE Transactions on Software Engineering, SE-5(5), pp. 440-452.

[DIS94] "The DIS Vision, A Map to the Future of Distributed Simulation" Institute for Simulation & Training, Orlando FL, May 1994.

[DMSO96] Defense Modeling and Simulation Office, "HLA Time Management: Design Document," 1996.

[Fuji95] R. M. Fujimoto, "Parallel and Distributed Simulation," In 1995 Winter Simulation Conference Proceedings, pp. 118-125, December 1995.

[Fuji96] R. M. Fujimoto and R. M. Weatherly, "HLA Time Management and DIS," In 14th Workshop on Standards and Interoperability of Distributed Simulations, March 1996.

[Jeff85] D. R. Jefferson, Virtual Time, *ACM Transactions on Programming Languages and Systems*, 7(3): 404-425, July 1985.

[Jha94] V. Jha and R. Bagrodia, "A Unified Framework for Conservative and Optimistic Distributed Simulation," 1994 Workshop on Parallel and Distributed Simulation, pp. 12-19, July 1994.

[Lamp78] L. Lamport, Time, Clocks, and the Ordering of Events in a Distributed System, *Communications of the ACM*, 21(7): 558-565, July 1978.

[Wils94] A. L. Wilson and R. M. Weatherly, "The Aggregate Level Simulation Protocol: An Evolving System," In 1994 Winter Simulation Conference Proceedings, pp. 781-787, December 1994.