HLA TIME MANAGEMENT AND DIS

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

Keywords: Architecture, Interoperability, Simulation Time, Time Synchronization

ABSTRACT

The High Level Architecture (HLA) effort is viewed by many as the next generation for DIS. HLA encompasses a broad range of simulation applications including training, analysis, and test and evaluation of components and systems. A challenging aspect of the HLA concerns defining a single time management structure that not only supports a wide variety of federations (e.g., DIS, ALSP, hardware-in-the-loop simulations), but also supports interoperability among simulations using different local time management mechanisms. For example, a single federation execution may include both DIS simulations where interactions are based on the *real-time* arrival of simulation messages, and constructive simulations such as ALSP where events must be processed according to *logical time* timestamp order to ensure cause-and-effect relationships are correctly reproduced by the simulation.

This paper describes the time management services that have been proposed for the HLA. These services include different categories of message transportation reliability and ordering, and mechanisms for controlling time advances. The ramifications of these time advance services on DIS are discussed.

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) [Wilson94]. 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 concept consist of three parts: 1) a set of rules that govern 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 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 several categories of services, as 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 will address the services concerning time management. The challenge to the RTI is to bring together, in a general and extensible way, the time management mechanisms represented by several disparate communities including DIS, ALSP, and test and evaluation. Below, we briefly review key concepts prevalent in DIS and ALSP before describing the HLA.

Distributed Interactive Simulation

"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 may include (1) virtual 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 or aircraft. Each simulation advances in time according to a (local) hardware clock. Any changes in state that might affect other simulations (e.g., firing a weapon) are broadcast to all other simulations. Each simulation is responsible for determining what information is relevant to the entities it is modeling, and what information can be discarded. The sender transmits perfectly accurate "ground truth" information, and it is the receiving simulation's responsibility to degrade the information in accordance with environmental effects (e.g., smoke) and/or sensor limitations.

DIS simulations typically process incoming state updates and interactions in the order that they are received. Because the DIS infrastructure does not provide message ordering services, different simulations may perceive the same set of updates in different orders. DIS exercises typically utilize unreliable message transmission services, sacrificing reliability in favor of lower communication latency. Because simulations generate state update messages "as they occur" at simulation time "now," simulations are constantly receiving "old" information, with its age determined by the latency in generating, transmitting, and receiving the information at the destination simulation. For information such as position updates, the receiving simulation can compensate for this latency by extrapolating forward (e.g., using dead-reckoning models), effectively assigning a new timestamp to reflect the time at which the information was received. Some temporal inaccuracies are acceptable in DIS because it has been found that in most cases, human perception cannot detect inaccuracies of up to 100 milliseconds.

Aggregate Level Simulation Protocol

ALSP was designed to extend the DIS concept, and focused to a large extent 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 timestamp order. ALSP currently uses a Chandy/Misra/Bryant style null message protocol to synchronize the distributed simulation [Chan79].

Parallel Discrete Event Simulation

ALSP is one instance of a broader body of research concerned with fast execution of discrete event simulation programs on parallel and LAN connected distributed computers. Work in the parallel discrete event simulation (PDES) community has been primarily concerned with reducing the execution time of as-fast-as-possible (as opposed to real-time) simulations where causality is maintained by ensuring that each simulation processes events in timestamp order, thereby eliminating the possibility of any event affecting those in the past. Because events are processed in timestamp order, PDES protocols are able to guarantee repeatable executions, i.e., successive executions of the same simulation program, using the same input data, can be made to generate exactly the same results. PDES research typically assumes reliable communications.

PDES protocols to ensure that events (state updates and interactions) are processed in timestamp order are generally classified as either *conservative* or *optimistic*. Conservative protocols *prevent* any simulation entity from ever processing events out of timestamp order. The Chandy/Misra/Bryant null message protocol used in ALSP is a well known example of a conservative mechanism [Chan79]. Optimistic protocols use a *detection and recovery* mechanism where out of order execution is detected at runtime, and a rollback mechanism is used to recover. Jefferson's Time Warp mechanism is the most well known optimistic protocol [Jeff85]. See [Fuji90] for an introduction to PDES research.

OVERVIEW OF THE HLA

Real-world entities are modeled in the HLA by objects. The HLA does not assume the use of object-oriented programming languages, however. Each object contains an identity that distinguishes it from other objects, state for the object, and a behavior description that specifies how an 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 that in another object.

Each object attribute has an owner that is responsible for generating updates to the value of that attribute (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. Any number of other federates may *subscribe* to receive updates to attributes as they are produced by the owner.

The HLA includes a non-runtime and a runtime component. The non-runtime component specifies the object model used by the federation. This includes the set of objects chosen to represent the real world, the attributes, associations, and interactions of these objects, the level of detail at which the objects represent the world (including spatial and temporal resolution), and the key models and algorithms (e.g., for dead-reckoning) that are to be used.

Each simulation must define a *simulation object model (SOM)* that identifies the objects used to model real-world entities in the simulation, and specifies the public attributes, the attributes whose ownership may be transferred, and those attributes whose value must be imported. Using the SOMs for the simulations that are included in a particular federation, a *federation object model (FOM)* must then be developed that describes the common object model used by all simulations in the federation. The FOM specifies all shared information (objects, attributes, associations, and interactions) for a particular federation. The HLA includes object model templates (OMTs) to provide a standard, tabular format for specifying objects, attributes, and the relationships among them.

The runtime component defines a set of services invoked by simulations 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 simulations to join or resign from existing federations, and to pause, checkpoint, and resume a federation execution.
- *Declaration management.* These services provide the means for simulations to establish their intent to publish object attributes and interactions, and to subscribe to updates and interactions produced by other simulations.
- *Object management.* These services allow simulations to create and delete object instances, and to produce and receive individual 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. Certain object management services are also discussed where they relate to time management.

INTEROPERABILITY ISSUES

Conceptually, the runtime infrastructure provides a base into which separately developed federates 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 simulations utilizing different internal time management mechanisms. Specifically, a single federation execution may include:

1. simulations with different message ordering requirements, e.g., DIS simulations with no ordering requirements and ALSP simulations that require messages to be delivered in timestamp order,
2. simulations using different internal time flow mechanism, e.g., timestepped and event driven simulations,
3. real-time (or scaled real-time) simulations and as-fast-as-possible simulations,
4. parallel simulations executing on multiprocessor platforms using conservative (non-rollback-based) or optimistic (rollback-based) synchronization protocols, or

5. individual simulations using a mixture of message ordering and transportation services, e.g., training simulations with message ordering and reliable delivery for certain types of events (e.g., weapon detonations) intermixed with unordered, best-effort delivery for others.

Allowing different messages within a single simulation to utilize different categories of service enables gradual, evolutionary exploitation of previously unused HLA-TM services.

The HLA supports these capabilities provided federates adhere to certain requirements necessary to realize each service, e.g., lookahead (discussed later) is required to provide timestamp ordering of messages. Further, individual federates executing in conjunction with the RTI must deliver real-time performance in federations whose execution is paced by real-time clocks.

A central design goal that is used to achieve interoperability is *time management transparency*. This means the local time management mechanism used within each federate must *not* be visible to other federates. This is realized by developing a *single*, unifying approach to time management that provides a variety of services needed by disparate simulations. Different categories of simulations typically use only a subset of the RTI's full capability. Simulations need not explicitly indicate to the RTI the local time management mechanism being used. Each federation execution globally specifies a real-time scale factor to indicate the rate each simulation attempts to advance in time relative to wallclock time. As-fast-as-possible federations specify "infinity" as the scale factor.

ASSUMPTIONS AND DEFINITIONS

The time management system makes the following assumptions concerning the federation:

1. No common, global clock is assumed. At any instant in the execution, different federates may have advanced to different times. This will often be the case for federates that must coordinate time advances with other federates to adhere to causality constraints. Even in (scaled) real-time simulations, drift between hardware clocks could result in federates having different local clocks at an instant in the execution.
2. It is assumed an external, synchronized wall clock of some specified accuracy is available to both the RTI and individual federates.
3. Each event is assigned a timestamp that is determined by the federate generating the event. Other timestamps are assigned to events to facilitate real-time simulations, however, the ordered message service provided by the RTI is based on this sender-assigned timestamp. A consequence of this assumption is that federates may generate (schedule) events with timestamps "in the future", i.e., timestamp larger than the federate's current time.
4. Federates may not generate events with timestamp "in the past," i.e., with timestamp smaller than the federate's current time.
5. Federates need *not* generate events in timestamp order. For example, a federate may first generate an event with timestamp 10, then later generate a new event with timestamp 8.

The classes of simulations supported by the HLA may be characterized according to two dimensions. The first dimension distinguishes between *constrained* simulations where there is a fixed relationship between the rate of advance of the simulation and wallclock time, and *unconstrained* simulations where there is no such relationship. Here, constrained simulations are also referred to as *(scaled) real-time simulations.* Unconstrained simulations are also referred to as *as-fast-as-possible* simulations. The second dimension concerns whether the simulations coordinate their advances in time with one another. *Coordinated* simulations use a protocol (e.g., Chandy/Misra/Bryant) to ensure message ordering constraints are met, while *independent* simulations advance local time independent of other simulations, usually paced by wallclock time. DIS simulations use an independent time advance mechanism. ALSP uses coordinated time advance. DIS executions are always constrained (i.e., real-time), but ALSP executions may be either constrained, or unconstrained.

The HLA time management structure distinguishes between two distinct notions of time: *(scaled) wallclock* time is a federate's measurement of the true global time, typically derived from a hardware clock. A scale factor is used to expand/compress time, e.g., a scale factor of two indicates the federate is running twice as

fast as real-time. Advances in wallclock time cannot be controlled by the federate. *Logical time* refers to a federate controlled time value. Logical time is what is commonly referred to as "simulation time" in the classical discrete event simulation literature (typically, as-fast-as-possible simulations), and is used for coordinated time advancement. If a federate advances its logical time to T, then it has declared that it has simulated all entities under its control up to time T. Operationally, logical time is advanced using the time advance services that are defined in the RTI. The *current time* of a federate is defined as the minimum of its wallclock and logical times.

An *event* refers to an attribute update, interaction, instantiate, or delete action performed by a federate. The corresponding primitive must be invoked to inform the RTI of an event. Operationally, an event causes messages to be transmitted to federates that have declared an interest in that event. Here, the terminology "delivering an event" to a federate is sometimes used as shorthand to mean "delivering a message that contains information concerning an event." It should be noted that messages may be used for other purposes besides transporting event information, however.

TIME MANAGEMENT IN THE HLA

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., attribute updates and interactions) to individual federates because such information is timestamped to indicate when the information is valid. For example, federates may require that no information be received "in the federate's past," i.e., with timestamp less than the current time of the federate. Thus, the time management services supported in the HLA 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 simulations to request advances in logical time. A simple protocol is provided to enable a federate to control the flow of attribute updates and interaction requests to that federate.

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 simulation. 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 described in detail later.

Time advance primitives provide the means for federates to coordinate their time advances with the timestamp of incoming information, if this is necessary. The time advance mechanism in the RTI must accommodate both real-time, scaled real-time, and as-fast-as-possible executions. The HLA time advance mechanisms are described later.

MESSAGE ORDERING

Central to the HLA time management services are mechanisms to order messages that are passed to federates. A variety of services are provided to support interoperability among federates with diverse requirements. Four ordering mechanisms are currently specified in the HLA: receive, priority, causal, and timestamp order. These provide, in turn, increased functionality but at increased cost.

Each federate may intermix different message ordering services for different types of information within a single federation execution. For example, position updates where reliable delivery and ordering are not important may utilize a best effort, receive order category of service. These messages may be intermixed with messages for ordnance detonation events utilizing reliable, timestamp ordered delivery.

Receive Order

This is the most straightforward, lowest latency ordering mechanism. 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.

Receive order should be utilized by applications where minimizing communication latency is more important than adhering to message orderings that guarantee causality. It is anticipated that simulations with hard real-time constraints (e.g., hardware-in-the-loop) will use this service. Similarly, many DIS federations are expected to use this mechanism, at least in the near term.

Priority Order

Incoming messages are placed in a priority queue, with the message timestamp used to specify its priority. Messages are passed to the federate lowest timestamp first. In other words, the RTI attempts to deliver messages in timestamp order based on the local information available to the RTI when the message is delivered. However, a message could later arrive and be delivered to the federate that has a timestamp smaller than one that has already been delivered. Further, this service does not prevent a message from being delivered in a federate in its "past" (timestamp less than the federate's current time). While this service does not guarantee timestamp ordered delivery, it is less costly in terms of latency and synchronization overhead than the service guaranteeing timestamp ordered delivery. This ordering should *not* be used if timestamp ordering of messages is essential to the correct operation of the federate.

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.

Messages using receive or priority order are available for delivery to the federate as soon as the message has been received. By contrast, the RTI may buffer messages using causal or timestamp order until it can guarantee the desired ordering properties.

Timestamp Order

Messages utilizing this service will be delivered to federates in timestamp order. To accomplish this task, the RTI will hold incoming messages in its internal queues until it can guarantee that no other timestamp ordered message containing a smaller timestamp will later be received. Simulations are *not* constrained to generate messages in timestamp order. A conservative PDES style synchronization protocol is used to implement this service.

In addition to delivering messages in timestamp order, the RTI also ensures that no event is delivered to a federate "in its past," i.e., no message is delivered that contains a timestamp less than the federate's current time. This is accomplished by forcing the federate to explicitly request advances in logical time using the RTI's time advance services. The RTI will not provide a "grant" to the time advance request until it can guarantee that no messages containing a timestamp smaller than the time of the grant will later be received.

This timestamp order service is required for classical discrete event simulations (e.g., constructive simulations) where timestamp order event processing is the norm. Federates that do not traditionally utilize timestamp ordered event processing, e.g., DIS simulations, but could be enhanced with strong event ordering properties for certain types of information may also utilize this service, while continuing to use less costly event ordering services for other information where order is less critical.

An important feature of the timestamp order service is that 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. This eliminates certain temporal anomalies that might otherwise occur when different simulations perceive different orderings of events. The RTI provides a consistent tie breaking mechanism so events containing identical timestamps will be delivered to different simulations in the same order.

Causal Order

Like the timestamp order service, the causal order service ensures that messages are delivered to federates in an order that is consistent with before

and after relationships of the events represented by these messages. For example, firing a weapon must occur before the target is destroyed. Unlike the receive and priority ordered services where varying communication latencies could cause the message for the destroy event to be delivered before the message for the fire event, the causal and timestamped ordered services guarantee that that if one event "happens before" a second, then all federates will receive messages for the first prior to the messages for the second.

The fundamental difference between the causal and timestamp ordered services is concerned with their respective definition of the "happens before" relationship. In the timestamped order service, an event is said to happen before another event if it has a smaller timestamp than the second, following one's intuitive notion of event ordering in physical systems. The federation can precisely specify which events happen before which other events by assigning appropriate timestamps, e.g., the "fire" event might be assigned a timestamp of 102, and the "destroy" event a timestamp of 103 to specify that "the fire event happens before the destroy event."

In the causal ordered service, the "happens before" relationship is that defined by Lamport [Lamp78]. The execution of each federate can be viewed as an ordered sequence "actions" (e.g., execution of a single machine instruction can be viewed as an action). Two specific actions of particular interest are sending and receiving a message. Lamport defines the happens before relationship as follows: (i) if actions A and B occur in the same federate, and A appears before B in the ordered sequence of actions within that federate, then A happens before B (ii) if A is the action sending a message to another federate, and B is the action receiving the same message in the second federate, then A happens before B, (iii) if A happens before B, and B happens before C, then A happens before C (transitivity). The happens before relationship can be easily extended to include messages: (i) if a message X is sent by a federate before the same federate sends another message Y, then X happens before Y, (ii) if X is received by a federate before that federate sends another message Y, then X happens before Y, (iii) if X happens before Y, and Y happens before a third message Z, then X happens before Z. An event in the HLA is said to causally precede another event if the first event happens before the second, using the "happens before" relationship defined above.

The HLA causal event ordered service guarantees that if an event E causally precedes another event F and messages for both events are delivered to a federate, then the message for E will be delivered to that federate before the message for F.

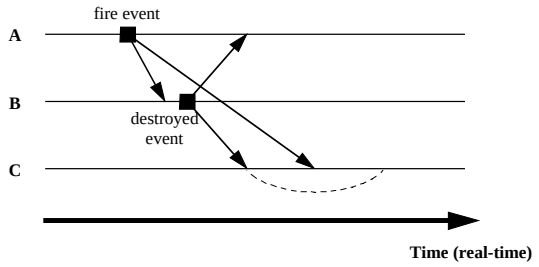

Figure 1. Scenario demonstrating causal event ordering. The RTI delays delivery of the first message received by C.

For example, consider the example depicted in Figure 1 where federate A fires a missile at federate B, destroying an entity in that federate, and a third federate C observes this exchange. The fire event in federate A generates messages to federates B and C. Upon receiving this event, federate B generates a state update event indicating that an entity it contains has been destroyed. In the scenario depicted in this figure, the "destroy" message reaches C before the "fire" message. Because the "fire" event causally precedes the "destroy" event, the causal event ordering service guarantees that the fire event is delivered to C prior to the destroyed event. The RTI will delay delivery of the message for the "destroy" event until after it receives and delivers the message for the "fire" event to C. The receive and priority ordered services do not guarantee causal ordering, so they could deliver the message for the destroy event before the message for the fire event in this scenario.

In the basic causal ordered service, messages corresponding to events that are *not* causally related (referred to as concurrent events) may be delivered to federates in any order. A variation on causal ordering is to guarantee that all federates receive messages for concurrent events *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. Algorithms for implementing CATOCS have

been developed and implemented (e.g., see [Birm91]).

Contrasting Timestamp and Causal Order

It is important to understand the differences between timestamp and causal order in order to determine which is appropriate for specific types of information. Timestamp order provides more stringent ordering services than causal ordering. Both the timestamp and causal ordering guarantee that messages for causally related events are delivered to federates in the order dictated by the causal happens before relationship. However, the ordering of concurrent events in the causal ordered service, even if CATOCS is used, is non-deterministic because it depends on latencies within the communication network. Thus, causal ordering, with or without total ordering, is *not* sufficient to produce repeatable results. Timestamp ordering must be used if this is a requirement.

Further, while causal ordering is sufficient to avoid certain anomalies (e.g., receiving a message for a tank destroyed event before the message for the event indicating the tank has been fired upon), it is not sufficient in other situations. Specifically, causal order is not sufficient if ordering relationships among *concurrent* events are important, or if there are "hidden dependencies" between events, as elaborated upon below.

Ordering Concurrent Events. Consider a simulation that includes three federates, each representing a tank. Suppose tank A has orders to fire upon the first target to come within range of its cannon. It may happen that tank B comes within range before tank C. However, because state updates by B and C are concurrent events, causal ordering does not guarantee that B's state update message will reach A before C's update message. This could cause A to incorrectly fire upon C. If capturing this behavior is important to the objectives of the federation, the timestamp order service should be used rather than the causal order service. Timestamp order will produce correct results because B's update indicating it enters A's range will have a smaller timestamp than C's update, so B's message will be delivered first.

Hidden Dependencies. Consider a battle in a military campaign that is staged as a timed sequence of actions, e.g., a diversion might be initiated by one unit at time 100, followed by initiation of the actual attack by another unit at time 150. It is clear that the commanders planning the operation staged its execution so that the diversion occurs before the main attack. However, CATOCS does not guarantee that messages corresponding to the diversion reach federates representing the opposing force before messages corresponding to the main attack! The problem is that all message ordering in CATOCS is based *only* on messages passed between federates, so semantic relationships between events are not visible to the RTI. Again, this is an instance where timestamp ordering must be used to ensure federates receive messages for events in the correct time sequence.

The principal advantage of causal ordering relative to timestamp order is that it does not require specification of lookahead (discussed later). Thus, causal order may provide an acceptable alternative to timestamp ordering for simulations with little lookahead where use of optimistic (rollback-based) event processing techniques is not considered viable. Further, it is anticipated that in most implementations, causal message order (at least the basic service without total ordering) will typically yield lower communication latency and require less bandwidth to implement than timestamp order.

Lookahead

The timestamp order service requires specification of a quantity called lookahead. To motivate the need for lookahead, consider a federation where timestamp order is specified for all communications. Consider the federate with the *smallest* logical time at some instant in the execution. Let this federate have a logical time of T. This federate could generate events relevant to every other participant in the federation with a timestamp of T. This implies the RTI cannot deliver any message with timestamp larger than T to any federate. In turn, this implies no federate can advance its logical clock beyond T because it is then prone to receiving an event in its past.

This is a well-known problem that has been widely studied in the PDES community. The two principal approaches to circumventing it are:

- use the notion of lookahead, described below, to define conservative protocols that prevent out-of-order delivery of messages, or
- use optimistic synchronization techniques that allow messages to be delivered out of timestamp order, and use a rollback mechanism within the federate to recover from errors introduced from out-of-order delivery.

HLA-TM supports both of these approaches. Optimistic synchronization is beyond the scope of this paper, but is described elsewhere [Fuji96]. Lookahead is elaborated upon next.

If one were to mandate that *no simulation may schedule an event with timestamp less than the federate's current time plus a value L,* then the RTI can allow concurrent delivery and processing of messages in a time window L time units wide beginning at the minimum logical time of any simulation in the federation. This value L is referred to as the lookahead for the simulation because a federate must be able to "look ahead" L time units into the future, or in other words, predict attribute updates and interactions at least L time units "ahead of time". Lookahead may, in general, be difficult to incorporate into certain classes of simulations, but nevertheless is very important for simulations requiring guaranteed message ordering services to achieve acceptable performance.

Lookahead is clearly very intimately related to details of the simulation model, and thus cannot be determined automatically by the RTI. Some examples of where lookahead may be derived are described below.

- *Physical limitations concerning how quickly one federate can react to an external event.* Suppose the minimum amount of time for a tank to respond to an operator's command (e.g., to fire an ordnance) is 500 milliseconds. This means the simulation can guarantee that it will not schedule the results of any new operator actions until at least 500 milliseconds into the future, providing a lookahead of this amount.
- *Physical limitations concerning how quickly one simulation can affect a second simulation.* Suppose two tanks are ten miles apart, and there is also a maximum speed of a projectile fired from one tank to another. These constraints place a lower bound on how much time must elapse for the first tank to affect the second. Thus, events such as a projectile exploding at the second tank can be scheduled into the future, providing some degree of lookahead.
- *Tolerance to temporal inaccuracies.* Suppose a simulation produces an event at time T, but the receiver of the message for that event cannot distinguish between the event occurring at time T and T+100 milliseconds (e.g., in a training simulation, it might be the case that a muzzle flash occurring at time T is indistinguishable from one at time T+100 milliseconds). Then, the simulation may schedule events 100 milliseconds into the future, providing a lookahead of this amount.
- *Time stepped simulations:* In a time stepped simulation, the lookahead is normally the size of the time step. This is because a simulation can only schedule events into the next time step (or later), but not into the current time step.
- *Non-preemptive behavior.* Suppose a tank is moving north at 30 miles per hour, and nothing in the federation model could cause it to change any events produced by the tank over the next ten minutes. These events could therefore be scheduled immediately, resulting in a lookahead of 10 minutes.
- *Precomputing simulation activities.* If the events produced by a simulation over the next L units of time do not depend on external events, but only depend on internal computations, these computations can be performed in advance, enhancing lookahead. For example, if the time until the next interaction of a simulation with another simulation is drawn from a random number generator, the generator can be sampled ahead of time, and the computed value can be used to derive the simulation's lookahead for this time instant.

Lookahead can change dynamically during the simulation. 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 timestamp ordering) with timestamp 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 timestamp less than C+L are produced.

The RTI requires each simulation to specify lookahead information if any events utilizing the guaranteed event ordering service are generated. Care must be taken in developing the federate to maximize lookahead, as this can significantly affect performance. A single lookahead value is designated by each federate. This value may change at runtime, but reductions in lookahead do not take effect immediately, as noted above.

TRANSPORTATION SERVICES

The two reliability of delivery categories (reliable and best effort) and four message ordering categories (receive, priority, causal, and timestamp ordered) result in eight different transportation categories. At present, five categories of transportation service are expected to be the most useful:

- Category BRec: (aka Category I) best effort message delivery, receive order.
- Category RRec: (aka Category II) reliable message delivery, receive order.
- Category BP: (aka Category III) best effort message delivery, priority order.
- Category RTS: (aka Category IV) reliable message delivery, timestamp order.
- Category RCO (aka Category V): reliable message delivery, causal order.

The category name (BRec, RRec, etc.) is derived with the first letter indicating best effort delivery (B) or reliable delivery (R), and the remaining letters indicating the type of ordering (Rec for receive order, P for priority order, TS for timestamp order, and CO for causal order).

Categories providing increased reliability and/or functionality will only do so at increased cost in terms of latency of message delivery and the network bandwidth required to support the service. Therefore, simulations should always use the least costly transportation service adequate for its objectives. Other categories of transportation service, specifically, different types of message ordering, may be defined in the future.

Different approaches may be taken regarding how the category of service is specified. For instance:

1. Each federate might specify the category of service for the messages it receives when it subscribes for that information, thereby allowing different federates to view incoming information in a way that is most appropriate for that federate. This would be specified when the federate subscribes to the information.
2. Each federate may select the most appropriate category of service on each message send, thereby allowing federates within a single execution to interleave generation of messages using different categories of service in arbitrary ways. This enables federates to select the appropriate category of service based on the type of information that is being transmitted.
3. Each federate may select the category of transportation service when it announces it is publishing that information.

The first approach is the most desirable from a modeling perspective because it enables each federate to treat incoming information in the manner most appropriate to the way it intends to use that information. Therefore, this is the approach specified by the HLA. However, initial versions of the RTI will not fully implement this approach. It is anticipated that initial implementations will utilize an approach closer to the third method described above.

Scheduling and Retracting Events

These are object management services. The **Update Attribute Values** and **Send Interaction** services are used to schedule events. Invocation of these services typically results in the generation of one or more messages transmitted to federates subscribing to the requested information.

Event retraction refers to the ability of a federate to retract (sometimes called cancel or unscheduled, but “cancel” refers to a mechanism used in optimistic federates) a previously scheduled event. This is a common discrete-event simulation primitive often used to model interrupts and other preemptive behaviors. The **Update Attribute Values** and **Send Interaction** RTI services return a handle for the event that is used to specify the event that is to be retracted.

Event retraction is available for all categories of service. 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 event has already been forwarded to the federate or is delayed in the network), the retraction request is forwarded to the federate.

TIME ADVANCE SERVICES

The time advance services provide a means for the federate to control the advancement of its logical time. Time advance requests typically release new messages to the federate, as described below.

- **Time Advance Request (t):** Requests an advance of the federate's logical time to t, thereby releasing all incoming Category III, IV, and V messages to the federate with timestamp less than or equal to t, and all Category I and II messages. The messages are passed to the federate by the **Reflect Attribute Values** and **Receive Interaction** services provided by the federate, and invoked by the RTI. The request is completed by a **Time Advance Grant** as indicated below. By issuing this request, the federate is guaranteeing it will not generate any Category IV messages in the future with timestamp less than t plus the federate's lookahead.
- **Next Event Request (t, one or all):** Requests the message for the next Category IV event from the RTI, provided that event has a timestamp no greater than t. The second parameter specifies the number of events passed to the federate (from any category) as a result of invoking this service, i.e., either one or all available events are requested. If "all" is specified, all received category I and II events, and category III and V events with timestamp no greater than t are also passed to the federate in addition to (at most) one category IV event. If "one" is specified, at most one event from *any* category is delivered. The events are passed to the federate by the **Reflect Attribute Values** or **Receive Interaction** service. If "all" is specified, a **Time Advance Grant** completes this request and indicates to the federate that it has advanced its logical time to the timestamp of the single Category IV event that is delivered, if any, or to the time specified in the **Next Event Request** (i.e., t). If "one" is specified one service (**Reflect Attribute Values, Receive Interaction,** or **Time Advance Grant**) is invoked as the result of this request**.** If there are no messages for Category IV events with timestamp less than or equal to t, and no such messages will be received in the future, a **Time Advance Grant** is delivered to the federate without delivering any events. By issuing this request, the federate is guarantee that if it does not receive any additional events with timestamp less than t, the federate will not generate any Category IV messages in the future with timestamp less than t plus the federate's lookahead.

The following services provided by the federate are invoked by the RTI:

- **Time Advance Grant:** Invocation of this service indicates a prior request to advance logical time has been honored. Specifically, invocation of this service indicates either (1) the RTI has delivered all messages for Category IV events to the simulation with timestamp less than or equal to that specified in the **Time Advance Request**, or (2) there are no messages for Category IV events with timestamp less than or equal to the time specified in the last **Next Event Request** service invocation using the "all" option, or (3) there are no messages for Category IV events with timestamp less than or equal to the time specified in the last **Next Event Request** service invocation using the "one" option and no messages for events from any category available for delivery. In either case, the federate's logical time is advanced to that specified in the **Time Advance Request** or **Next Event Request**. If a category IV event is delivered when one event is requested via the **Next Event Request** service, logical time is advanced to the time of the last Category IV event that was delivered. The RTI will not deliver other Category IV events in the future with timestamp less than that of this advance.
- **Reflect Attribute Values:** The RTI will invoke this service to deliver a new reflected attribute value to the federate.
- **Receive Interaction:** The RTI will invoke this service to deliver a new interaction to the federate.

It is perhaps noteworthy that as defined above, **Time Advance Request**, **Next Event Request**, and **Time Advance Grant** only pertain to the

advancement of logical time. Real-time advances independent to the federate's execution (of course!). The relationship between logical and real-time is expanded upon next.

REAL-TIME SIMULATIONS

The relationship of the operation of the RTI to real time merits closer examination. Recall that the current time of the federate is defined as the minimum of two quantities:

- the scaled wallclock time, and
- the *logical time* of the simulation (advanced by the **Time Advance Request** and **Next Event Request** services).

As-fast-as-possible simulations used in a non-real-time (i.e., as-fast-as-possible) federation execution will set the scale factor for wallclock time to infinity, effectively setting wallclock time to infinity. This makes the current time equivalent to logical time. Real-time simulations that do not require coordinated time advance with other simulations, e.g., simulations using only real-time synchronization as in DIS, set the logical time clock to infinity, thereby making the current time equivalent to wallclock time. The case of real-time simulations requiring coordinated time advances (e.g., as-fast-as-possible distributed simulations operating in a real-time setting) is discussed next.

Real-time (constrained) simulations with coordinated time advances may use the RTI based on the following paradigm:

- The timestamp on each event denotes a deadline by which processing of that event should be completed.
- Real-time computations often utilize a "start time" with each event to denote the earliest real-time when processing of the corresponding message may begin. The RTI attempts to deliver the message to the simulation as soon as possible, consistent with ordering constraints, but does not explicitly utilize the notion of start times. If CT_i is the current time of federate *i*, and L_i is that federate's lookahead (the minimum time into the future that a federate may schedule a Category IV event), the RTI will not deliver a Category IV event to a federate with timestamp larger than *minimum*$(CT_i + L_i)$ where *i* is computed over *neighboring* federates, i.e., those federates that send messages to the federate receiving the event. In other words, assuming skew between hardware clocks is negligible, the maximum amount of time an event will be delivered before its deadline is the minimum lookahead among its neighboring simulations. If this constraint is insufficient and could result in processing the event "too soon," the federate will need to internally delay processing the event by (for instance) specifying a start time and using an internal scheduling mechanism to specify when the event should be processed.
- Federates attempt to "stay ahead" of real-time by processing events before their deadline, and scheduling events sufficiently far "into the future" (timestamp larger than the local clock) so that they can be transmitted and delivered to receiving federates before their deadline has passed.
- Logical time (controlled by the federate generated time advance requests and RTI generated grants) controls delivery of Category III, IV and V events to the federate. Ideally, each federate can advance logical time so that it remains ahead of wallclock time, causing the federate's time advances to be paced by wallclock time. So long as *minimum*$(CT_i + L_i)$ remains larger than the current time of the receiver, the RTI can deliver events to the federate prior to their timestamp. This is a necessary, but not sufficient, requirement to avoid missing deadlines. It is not sufficient because the federate may not be able to process the events before the deadline has expired. In general, a more in-depth real-time analysis of event computations is required to guarantee that deadlines are not missed (e.g., for federations with hard real-time constraints).
- If logical time does *not* keep up with wallclock time, either because of delays in the RTI, federate, or the communication network, one or more federates will fall "behind" relative to wallclock time. The RTI will delay delivery of category IV events in order to ensure event ordering guarantees are met. This will, in turn, slow the granting of time advances of other federates, and could eventually cause many federates using coordinated time advance to proceed slower than real-time. In this case, the federation is

paced by logical time advances, which are advancing slower than real-time. This will be problematic for certain federations. Mechanisms to address this question, e.g., by relaxing event ordering constraints, are currently under investigation.

In general, it is clear that if an event is not delivered to the federate prior to its timestamp, it will be impossible to process it before its deadline. Thus certain real-time constraints must be met for the federation to operate in real-time. A necessary, though not sufficient constraint for real-time execution without missed deadlines is:

E_{TS} > T(real[sender]) + WC(max-error) + Comm(max-latency) + HLA-TM-delay

where:

- E_{TS} is the timestamp assigned to the event.

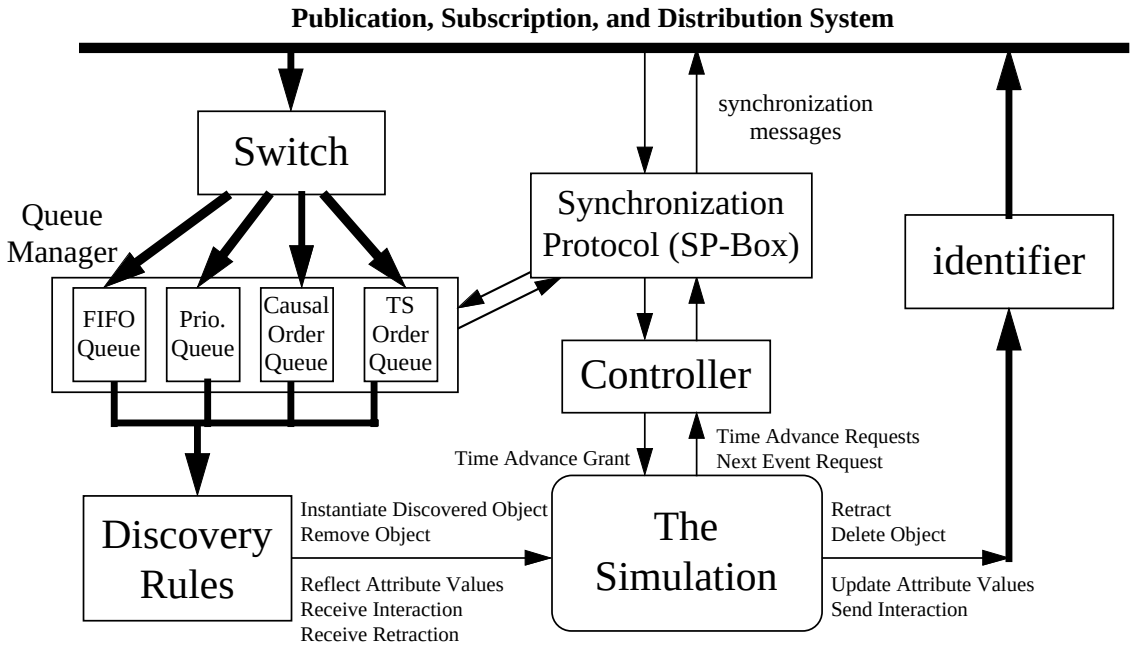

Figure 2. Logical flow of messages in the RTI.

- T(real[sender]) is the local wallclock time when the sending federate invokes the **Update Attribute Values** or **Send Interaction** service to pass the event to the RTI.
- WC(max-error) is the maximum difference between clocks in the federation, e.g., due to skew in the hardware clocks of different simulations.
- Comm(max-latency) is the maximum latency incurred in the communication network.
- HLA-TM-delay is the time the RTI must hold the message before delivering it to the federate. This includes other time-overheads associated with delivering a message to the application as well as the time the RTI must hold the message until event ordering can be guaranteed, for Category IV events.

In short, the federate needs to send the message far enough into the future so it can be received before its timestamp.

RTI STRUCTURE

The logical flow of messages within the RTI at a single site is depicted in Figure 2. Each federate receives a stream of incoming state update and interactions in accordance with the objects and attribute values to which the federate has subscribed. Each message has a category of service associated with it that specifies the reliability of the transport mechanism (reliable or best effort) and the event ordering to be used. The message is routed to an appropriate queue based on the type of ordering, as depicted by the "switch" module in Figure 2. Messages that have become available for delivery to the federate are passed to the discovery rules module where

filtering may be performed before they are passed to the simulation.

The synchronization protocol module in Figure 2 is responsible for implementing the more complex event ordering services (causal and timestamp order). The controller module handles time advance requests. Outgoing messages are sent through the identifier module that adds header information to messages before sending them onto the network.

CONCLUSIONS

The principal impacts of the HLA time management services in the DIS community are:

- The HLA time management services provide a structure to support integration of simulations with strict causality requirements (e.g., ALSP) with other simulations with minimal needs for event ordering (e.g., DIS).
- The transportation services range from basic, unordered communication services not unlike those used in DIS today, to services providing reliable message delivery and/or timestamp ordered delivery of messages to enable federations to faithfully reproduce cause-and-effect relationships in the systems they model.
- Different message delivery services may be intermixed within a single federation execution, allowing federates to utilize the category of service best suited for the type of information that is to be exchanged.
- Use of multiple categories of service enables evolutionary development of federations where new capabilities (e.g., higher reliability and consistent ordering) are gradually incorporated into the federation over time.

The HLA time management structure is perhaps unique in that it brings together extensive bodies of research from the DIS, ALSP, PDES, and distributed operating systems communities into one common framework for distributed simulation applications. Definition of HLA-TM services is an evolving process that will continue as more experience is gained in implementing different versions of the RTI.

ACKNOWLEDGMENTS

The time management approach used in the HLA is the result of the collective efforts of many individuals, including the members of the HLA time management working group. Other individuals that contributed 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 and suggested the example depicted in Figure 1. Richard Fujimoto's work as chair of the HLA Time Management group was funded by the Defense Modeling and Simulation Office (DMSO).

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): 440-452, September 1979.

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

[Fuji90] R. M. Fujimoto, Parallel Discrete Event Simulation, *Communications of the ACM*, 33(10): 30-53, October 1990.

[Fuji96] R. M. Fujimoto and R. M. Weatherly, Time Management in the DoD High Level Architecture, *10th Workshop on Parallel and Distributed Simulation*, May 1996.

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

[Lamp78] L. Lamport, Time, Clocks, and the Ordering of Events in a Distributed System,

Communications of the ACM, 21(7): 558-565, July 1978.

[Wilson94] 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.