

# High Level Architecture

## Module 2

### Advanced Topics

---



Roy Crosbie  
John Zenor  
Simon M. Goberstein

California State University, Chico

# **High Level Architecture**

## **Module 2**

### **Advanced Topics**

---

# **Time Management 1:**

## **Fundamentals and Design Philosophy**



California State University, Chico

9/16/99

2

# **Time Management in the HLA:**

## **1. Fundamentals and Design Philosophy**

---

- This lesson and the next one are based on R.M. Fujimoto's paper "Time Management in the High Level Architecture", *SIMULATION* **71:6**, 1998, 388-400



## Time in Distributed Simulations

- **Physical time**
  - Time in the physical system, that is, the system being modeled by the simulation
- **Simulation time**
  - The simulator's representation of physical time
- **Wallclock time**
  - Time when the simulator is executed



### **Example.**

In a simulation of the attack on Pearl Harbor, **physical time** might extend from midnight until 6:00 p.m. on December 7, 1941. In that case, **simulation time** might be represented as a double precision floating point number that can hold values in the interval  $[0.0, 18.0]$  where a unit of simulation time corresponds to an hour of physical time. If the Pearl Harbor simulation required four and a half hours to execute and if it were executed in the afternoon of May 7, 1999, **wallclock time** might extend from 13:00 until 17:30 on that day.

## Simulations Supported by the HLA

- **Scaled real-time simulations**
  - Have a linear relation  $T = S * W$  where  $W$  is a duration in wallclock time,  $T$  the corresponding duration in simulation time, and  $S$  is a scale factor
  - If  $S=1$ , we have a **real-time simulation**
- **As-fast-as-possible simulations**
  - When one attempts to complete the simulation as quickly as possible, so execution is **not** paced to have a direct relationship to wallclock time



### **Example 1.**

A training simulation where humans are embedded into a computer-generated virtual environment must be paced so that the environment appears realistic to the trainees. This would be an example of a **real-time simulation**.

### **Example 2.**

The simulator may run faster than wallclock time early in the execution, and it might run slower than wallclock time during other parts of the execution. This would be an example of an **as-fast-as-possible simulation**. Analytic traffic simulations frequently fall into this category.

## The Need for Time Management

---

- A simulator may incorrectly reproduce temporal aspects of the real world system being modeled
- Repeated executions of the simulation with the same initial state and external inputs might give entirely different results



1) In the simulated world, there might be delays that depend on the quantities having nothing to do with the simulation model, e.g., delays encountered by messages as they travel through a network. This can lead to anomalies such as *the cause appearing to happen after the effect*.

2) If no precautions are taken, it is possible that different simulators will receive messages for the same set of events in different orders. If the behavior of each simulator depends on the ordering of events, this can lead to other inconsistencies in the simulation execution.

3) It is sometimes important that repeated executions of the simulation yield the same results. However, the order in which messages are delivered to each simulator (i.e., federate) depends on such properties as communication delays in the network that may change from one execution to the next. This might lead to the situation when repeated executions of the simulation with the same initial state and external conditions may produce entirely different results.

4) The importance of correctly reproducing temporal relationships in the simulation model depends, of course, on the simulation application. This is very important in many analytic simulations because an impossible event ordering may not have been anticipated by the model developers which could cause a simulator to fail. On the other hand, in training applications, non-causal orderings of events may not be perceptible to human participants, and even if they are, they often can be just ignored without major consequences.

## Time Stamps

---

- Temporal anomalies are eliminated by assigning a **time stamp** (in simulation time) to each event and ensuring that events are delivered to the federate in **time-stamp order (TSO)**
- The time management services ensure a federate will not receive an event in its past, i.e., an event with time stamp less than its current federate time



The “cause” will always be assigned a smaller time stamp than any “effect”. This guarantees that causal relationships will be correctly reproduced by the simulation model. It should be noted that both time-stamp order requirement and the constraint that no events in a federate’s past are delivered to the federate at the present moment, can be relaxed by those federates that do not need them.

## Logical Time

---

- **Federate time** is the current simulation time of a federate
- **Logical time** is how federate time is called for federates that can send or receive TSO messages to distinguish these federates from those deriving federate time directly from wallclock time
- During the execution of a federation, different federates may have different logical time values



## Design Rationale

---

- A **key goal** of the HLA is to foster **interoperability** and **reuse** of simulations. The time management services:
  - Facilitate **reuse** by being flexible to accommodate the wide range of internal time management mechanisms
  - Support **interoperability** by allowing simulations with different internal time management mechanisms to be combined in a single federation execution



An important design principle that is used to foster interoperability is **time management transparency** discussed next.

## Time Management Transparency

---

- This means that the local time management mechanism used within each federate must **not** be visible to other federates
- Federates do not explicitly indicate to the RTI the local time management mechanism being used, and can even change their local time-flow mechanism during the execution



Thus a federate using, for example, an event-oriented time-flow mechanism need not know whether the federate with which it is interacting is using an event-oriented or a time-stepped mechanism, etc.

## Time Management Mechanisms

---

- **Event-driven**
  - The federate processes events in time-stamp order
- **Time-stepped**
  - Each time advance of the federate is in fixed *time step*
- **Parallel discrete-event simulation**
  - Federates on multiprocessor systems are synchronized using a conservative or an optimistic protocol
- **Wallclock-time driven**
  - Federate time is derived directly from wallclock time



1) **Event-driven.** The federate processes both local events and those generated by other federates in time-stamp order, and federate time typically advances to the time stamp of each event as it is processed.

2) **Time-stepped.** Each time advance made by the federate is of a fixed duration of simulation time called a **time step**. The simulator does not advance to the next time step until all simulation activities associated with the current time step have been completed. *Activity scanning*, another time-flow mechanism sometimes used in discrete-event simulation, is essentially a variation of the time-stepped mechanism.

3) **Parallel discrete-event simulation.** Federates executing on multiprocessor systems may be synchronized internally. A synchronization protocol used for that purpose can be either *conservative* or *optimistic*. In a **conservative** protocol, each *logical process* within the federate must process its events in time-stamp order. Logical processes may have to wait until this condition can be guaranteed. **Optimistic** synchronization protocols allow logical processes to process events out of time-stamp order but provide a means to recover from such errors, typically through the use of a roll-back mechanism.

4) **Wallclock-time driven.** Wall-clock-time driven federates do not require that events be processed in time-stamp order. These simulations usually have hard and/or soft real-time constraints and so interactions with humans and/or physical devices occur in timely fashion.

## Implementation: RTI or Federates?

---

- Time management in any federation must be realized jointly by the federates and the RTI
  - **Key question:** what functionality should be implemented within the RTI, and what should be realized within individual federates?
  - The answer depends on the type of time management mechanism used by the simulator; it will be discussed separately for each mechanism



## Event-Driven Federates

---

- **Tasks to be accomplished:**
  - Merge the stream of “external” events with local ones to get one stream of time-stamp ordered events
  - Ensure the federate does not receive “past” events
- **How the HLA accomplishes these tasks:**
  - A time-stamp order message delivery service
  - A protocol where federates request advances in their federate times, and the RTI grants these requests if it can guarantee that no events with smaller time-stamps will arrive later



## Time-Stepped Federates

---

- **Main task to be accomplished:**
  - Must check when all events in the given time step produced by other federates have been received
- **How the HLA accomplishes this task:**
  - The HLA uses a variation of the protocol described in the previous slide (we omit the details)



A time-stepped federate can complete the computation for the current time step and advance to the next time step only when it can determine that all events in the current time step produced by all other federates have been received.

## Parallel Discrete-Event Simulations

---

- Federates with *conservative* synchronization have similar requirements as **event-driven** ones since each *logical process* within any such federate must process its events *in time-stamp order* (Slide 13)
- *Optimistic* synchronization protocols let *logical processes* process events *out of time-stamp order* and a modest amount of additional functionality is required to support parallel simulators that use optimistic time management



## Wallclock-Time Driven Federates

---

- These simulators use *entirely different* mechanisms (from those described above) for managing time:
  - **Clock synchronization algorithms** - to ensure that hardware clocks generating wallclock time values are properly synchronized
  - **Real-time scheduling** - to guarantee computations are completed by certain deadlines for timely interactions
  - **Time compensation** - to take communication latency into account
- They **must** be implemented **within the federate**



California State University, Chico

9/16/99

16

- 1) Time management based on wallclock time is used in virtual simulators for training and for hardware-in-the-loop test and evaluation applications.
- 2) Clock synchronization is needed because hardware clocks in different (and geographically distributed) processors drift relative to one another. This can lead to serious problems in the distributed simulation if differences are large.
- 3) Task deadlines and laxities are often used to schedule the computations so that the deadlines can be met and timely interactions with hardware devices and/or human participants is ensured.
- 4) Suppose, for example, that a simulator for a moving vehicle generates a message indicating its current position, speed, and direction of motion. If the message is received 100 milliseconds later, the recipient can extrapolate the current position of the vehicle using dead reckoning techniques in order to estimate the vehicle's position at the moment the message was received.

## Why Within the Federate, Not in RTI?

---

- Dependence on the federation objectives and details of the models
- The need for detailed information about the computations performed in the federate
- The need for information concerning the semantics of what is being simulated
- Most of the above is not available in the RTI



Unlike message ordering and time-stamping requirements that are largely independent of the goals of the federation and what is being simulated, specifics concerning the time management mechanisms described in the previous slide are highly dependent on the federation objectives and details of the models. For example, hardware clocks must often be more closely synchronized in hardware-in-the-loop test and evaluation applications compared to interactive training simulations. Scheduling algorithms require detailed information concerning the computations performed within the federate, and thus are not well suited for implementation within the RTI. Time compensation techniques require information concerning the semantics of what is being simulated. Such information is not available within the RTI.

## **Minimalist Approach of the HLA to Wallclock-Time Driven Federates**

---

- Principal requirements with respect to the RTI:
  - Minimal, predictable communication latency
  - Minimal computational and communication overhead imposed by the RTI
- These requirements are satisfied in the HLA by:
  - Providing services for messages to be delivered with minimal latency and little additional functionality for time management (time-stamp order and control over time advances are eliminated if they are not needed)

