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 2:
The Services and the Interface
Time Management in the HLA:
2. The Services and the Interface

• This lesson (as well as the previous one) are based on R.M. Fujimoto’s paper “Time Management in the High Level Architecture”, SIMULATION 71:6, 1998, 388-400
Key Components of the Time Management Services

- A time stamp order delivery service
- A protocol for simulations to advance their logical times

We are going to discuss these two components in some detail. Slides 5-9 are devoted to the first one of them while slides 10-15 are devoted to the second.
Incoming Messages

• There are two types of incoming messages:
  – Receive-order (RO) messages, and
  – Time-stamp-order (TSO) messages
Receive-Order Messages

- Are placed in a queue when they arrive
- Are immediately eligible for delivery to the federate
- Are ordered arbitrarily
- May, optionally, contain a time stamp value which is simply passed, uninterpreted by the RTI, to the destination federate
Time-Stamp-Order Messages

- Are assigned a time stamp by the generating federate
- Are placed in a queue within the RTI but are *not* eligible for delivery until the RTI can *guarantee* that
  - there are *no* other TSO messages with a *smaller* time stamp destined for that federate
  - *no* TSO messages from other federates with a *smaller* time stamp will arrive *later*

To ensure the last property (the guarantee that no TSO messages with a smaller time stamp will arrive later from other federates), the RTI must compute a lower bound on the time stamp of future messages it may receive from other federates. More will be said about this later.
Simultaneous Events

- Messages with identical time stamps, referred to as *simultaneous events*, are delivered to the federate in an arbitrary order.
- There are several different ways to order these messages in a particular fashion which often might be important to do.

In the next slide, we will discuss the ways to do that.
Ordering Simultaneous Events

• The federate receiving simultaneous events can buffer and order them itself
  – To support this, the RTI provides a means for specifying when the federate has received all simultaneous events with a given time-stamp value

• Federates may include tie-breaking fields in the message time stamp to order simultaneous events
  – This requires federation specification of the format and meaning of the time-stamp field and logical time values and duration

To support the latter, time values can be specified by the federation using an abstract data type. This allows the time representation to be tailored to meet the requirements of different federations. In addition to specifying the time-stamp format, which must be documented in the Object Model Template, the federation must also specify certain operators such as comparison of time-stamp values. This capability enables HLA federations to utilize techniques to deterministically order events and to facilitate repeatable executions.
Advancing Logical Time

- Federates must explicitly request that their logical times be advanced
- Advances do not take place until the RTI grants them explicitly
- This protocol for advancing logical time is central to the HLA time management services

We start describing the second portion of the time management component of the HLA which provides a mechanism to advance simulation time within each federate. Since the RTI must guarantee that a federate will not receive any TSO messages with a time stamp less than its current logical time, the federates cannot autonomously advance their logical times.
A Three-Step Time Management Cycle

• The federate invokes a time management service to request that its logical time be advanced
• The RTI delivers some number (possibly zero) of messages to the federate
• The RTI invokes a federate-defined procedure called **Time Advance Grant** to indicate that the federate’s logical time has been advanced

A message is delivered to the federate by the RTI invoking a federate-defined procedure (for example, **Reflect Attribute Values** to deliver new values for object attributes, or **Receive Interactions** to deliver interaction events, etc.)
Mechanisms for Advance Requests

• The two principal mechanisms for a federate to request its logical time to be advanced are the following services:
  – **Time Advance Request (TAR)** which is well suited for those federates that internally use a time-stepped mechanism
  – **Next-Event Request (NER)** which is the preferred primitive for event-driven federates

There is **no** restriction concerning what primitive can be invoked by which federate. Any federate can exclusively use TAR or NER, or it can intermix calls between the two.
Time Advance Request (T)

• The federate invokes this service to request its logical time to be advanced to T
  – All RO messages in the RTI’s internal queues, and all TSO messages with time stamps ≤ T, are delivered to the federate after TAR(T) has been invoked
  – If no other TSO messages with time stamps ≤ T are forthcoming (or will be generated by another federate later), the RTI calls the federate’s **Time Advance Grant** procedure with parameter T to indicate that the federate’s logical time has been advanced to T
Next-Event Request (T)

- An event-driven federate will typically invoke this service when it has completed all simulation activity at the current logical time and is ready to advance to a new time.
- The parameter T in NER(T) is the logical time to which the federate would like to advance if there are no other events from other federates with smaller time stamps.

Typically, T is the time stamp of the next event in the federate’s local set of pending events.
Invoking Next-Event Request (T)

- After the federate F invokes NER(T):
  - The RTI delivers all RO messages in its internal queue
  - If no present or future TSO message has a time stamp \( \leq T \), the RTI invokes the federate’s *Time Advance Grant* procedure indicating that the federate’s logical time has been advanced to \( T \)
  - Otherwise, the RTI will deliver the next smallest TSO message destined for F (with time stamp \( T' \leq T \)) and all other messages with time stamp \( T' \); then the RTI calls *Time Advance Grant* procedure with parameter \( T' \), and the logical time of F is advanced to \( T' \)
Interface Issues

• The interface is designed so that:
  – The RTI gets all the information necessary for efficient implementations of the time management primitives
  – The RTI implementers are not constrained to use a particular synchronization protocol
• One can view the interface as a contract between the federate and RTI concerning the time stamp of messages the federation might generate later in the execution

_Time-stamp ordering_ and _primitives to advance time_ form _central components_ of the HLA time management architecture. Additional primitives are provided to support federates executing on parallel processors using optimistic synchronization protocols. The _optimistic_ technique allows messages to be processed out of time-stamp order, but uses some mechanism to recover. Jefferson’s Time Wrap mechanism is the most well known optimistic synchronization protocol (D. Jefferson, “Virtual Time”, _ACM Transactions on Programming Languages and Systems_, Vol. 7, No. 3, pp. 404-425, 1985).
The Lower Bound on Time Stamps

- The key to implementing the time management services within the RTI is to compute the lower bound on time stamp (LBTS) for each federate.
- $\text{LBTS}_F$, the LBTS value computed for federate $F$, is a lower bound on the time stamps of messages that may be received and that are destined for that federate later in the execution.
How LBTS Is Used

- For a federate F, the RTI must ensure that:
  - TSO messages are delivered to F in time-stamp order
  - No message is delivered to F with a time stamp that is smaller than its current logical time

- Once LBTS<sub>F</sub> has been computed,
  - The RTI can deliver to F all TSO messages containing a time stamp less than LBTS<sub>F</sub>
  - If the RTI prevents F from advancing its logical time beyond LBTS<sub>F</sub>, it can guarantee that F will receive no messages in its past
Computing LBTS

- To compute LBTS, the RTI must consider:
  - The smallest time stamp of any TSO message any federate might generate in the future (the current logical time of a federate is one bound since no federate can generate a TSO message in its past)
  - The time stamps of messages within the RTI and the interconnection network
- The algorithms for computing LBTS are beyond the scope of this discussion
Federates’ Role in LBTS Computations

- The RTI must identify those federates that **must participate** in LBTS computation and those that **require the results** of the computation
- **Time-regulating** federates are the ones that are able to **generate** TSO messages, so they must be considered in the computation of LBTS values
- **Time-constrained** federates are those that may **receive** TSO messages, so they require the results of LBTS computations

In the HLA, each federate contains two Boolean flags indicating to the RTI its status with respect to LBTS computations: the **time-regulating** and **time-constrained** flags. The HLA time management services provide primitives for federates to turn on or to turn off their constrained and regulating flags. The next slide explains what happens when these flags are turned on or off.
Time-Constrained/-Regulating Flags

- If the federate’s *regulating/constrained* flag is *not* set when it *sends/receives* a TSO message, the RTI will convert it to an RO message.
- Federates with their *regulating/constrained* flag set can *send/receive* both TSO and RO messages.
- Federates would *not* normally set these flags unless they had specific requirements to operate using time management services.

The regulating/constrained flags are normally *not* set because the LBTS computations require a certain amount of computational and communication resources that should be utilized only if there are specific requirements to do that.
Lookahead

- **Assume:** F is a federate that can send and receive TSO messages and L is some number of time units
- **Require:** No simulation may generate an event with a time stamp less than F’s current time plus L
- If F is at logical time T, any event generated by F has a time stamp $\geq T+L$; this would allow another federate to advance its logical time to $T+L$
- L is referred to as the **lookahead** for federate F

The term “lookahead”, used here for L, stresses that federate F must be able to “look ahead” L time units into the future to predict what events it will generate, and schedule these events early. Lookahead may be difficult to incorporate into certain classes of simulations but it is very important for simulations that require guaranteed message ordering services to achieve acceptable performance. Lookahead is closely related to the details of the simulation model, so the RTI cannot automatically determine it. **Lookahead can change dynamically during the simulation but it cannot instantaneously be reduced.** Indeed, at each moment, a lookahead of L indicates to the RTI that the federate, using time-stamp ordering, will not generate any event with time stamps $< T+L$ where T is the federate’s current logical 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 stamps $< T+L$ are produced.** In the HLA, each federate declares (its own) single lookahead value which may change at runtime but **reductions in lookahead do not take effect immediately.**
Lookahead Examples

• Limits on a federate’s reaction time:
  – If a machine needs at least 100 milliseconds to respond to an operator’s command, a simulation can guarantee it won’t schedule the results of any new actions by an operator until at least 100 milliseconds into the future by providing a lookahead of this amount.

• Time-stepped simulations:
  – Such simulations can only schedule events into the next time step (or later) but not into the current time step, so the lookahead is usually the size of the time step.
Zero Lookahead - A Problem?!

- If federate F receives a Time Advance Grant to time T after invoking TAR(T) or NER(T), the RTI guarantees that F has received all events with time stamps equal to T.
- This is an important and useful guarantee.
- If F has a lookahead value of zero, this guarantee cannot be kept which creates a problem!
- Yet prohibiting zero lookahead is too restrictive, so what to do?

The problem arises because the federate that has received a Time Advance Grant to logical time T could now generate a new event with time stamp equal to T (=T+0); this, in turn, might result in a second event, also with time stamp T (=T+0), which is sent back to the original federate! Early versions of the time management services avoided this problem by requiring that all federates have strictly positive lookahead values. During the HLA prototyping phase, it became clear that prohibiting zero lookahead values was too restrictive.

To support zero lookahead federates without giving up the possibility that the federates could get a guarantee from the RTI that they had received all events at a given time stamp, two new services were added:

Next Event Request Available (NERA) and Time Advance Request Available (TARA). They will be discussed in the next slide.
Zero Lookahead - A Solution!

• Next Event Request Available (NERA), and Time Advance Request Available (TARA) are similar to NER and TAR respectively but:
  – When a Time Advance Grant to time T is issued to a federate F in response to a TARA/NERA service call, the RTI does not guarantee F has received all messages with time stamps = T, and F can generate new messages with time stamps = T (i.e., zero lookahead messages)
  – When a federate F receives a Time Advance Grant to time T as the result of a TAR or NER call, F is not allowed to send messages with time stamps ≤ T even if F’s lookahead is zero

This approach allows federates to generate zero lookahead messages but also gives the federate the ability to determine if it has received all messages with time stamps equal to the current logical time
Guarantees on the Time Stamps

- The constraints on time stamps of new messages discussed so far might not prevent a simulation from deadlocking under certain circumstances.
- The time management services include a number of additional guarantees made by each federate to the RTI, regarding time stamps of any new events the federate might generate in the future, which are discussed in Section 6.5 of the paper.
Optimistic Event Processing

- HLA federations may include both optimistic and conservative federates in a single execution
- Conservative federates not interested in utilizing optimistic processing techniques may completely ignore the optimistic time management services
- The optimistic time management services must enable optimistic execution among a collection of optimistic federates; various capabilities needed for that purpose are discussed in the paper