Time Management in the High Level Architecture

Roger McFarlane
School of Computer Science
McGill University
Montréal, Québec
CANADA

19 March 2003

References

- Fujimoto, R.M. 1998. “Time Management in the High Level Architecture.” *SIMULATION Special Issue on High Level Architecture*, vol. 71, no. 6, 388-400
- C. D. Carothers, R. M. Fujimoto, R. M. Weatherly, A. L. Wilson, “Design and Implementation of HLA Time Management in the RTI version F.0,” *Winter Simulation Conference*, December 1997.
- “HLA Time Management Design Document, version 1.0.” U.S. Department of Defence, August 1996.
- Fujimoto R. & Weatherly R. “HLA time management and DIS” *14th Workshop on Distributed Interactive Simulation*, March 1995.

Overview

- The Run-Time Infrastructure
- What is Time?
- Why Time Management?
- Design Overview
- Detailed Design
- Discussion

The Run-Time Infrastructure (RTI)

- A special purpose distributed operating system
- Provides services to interconnect federates
- Typically defines as:
  - A client library (linked to each federate)
  - A set of remote services

RTI Services (1)

- Federation Management
  - Services to create, delete, pause, checkpoint, and resume a federation execution
  - Allow federates to join or resign from an existing federation
- Declaration Management
  - Allow a federate to declare its intent to publish object attributes and interactions
  - Allow a federate to subscribe to updates and interactions published by other federates

RTI Services (2)

- Object Management
  - Allow federates to create and delete object instances
  - Allow federates to produce and receive individual attribute updates and interactions
- Ownership Management
  - Enable the transfer of attribute ownership (the right to modify and publish) during federation execution

RTI Services (3)

- Time Management
  - Coordinate the advancement of logical time
  - Maintain the relationship between logical time and wall-clock time
- Data Distribution Management
  - Efficient data transfer between federates
  - Run-time maintenance and application of publisher/subscriber lists

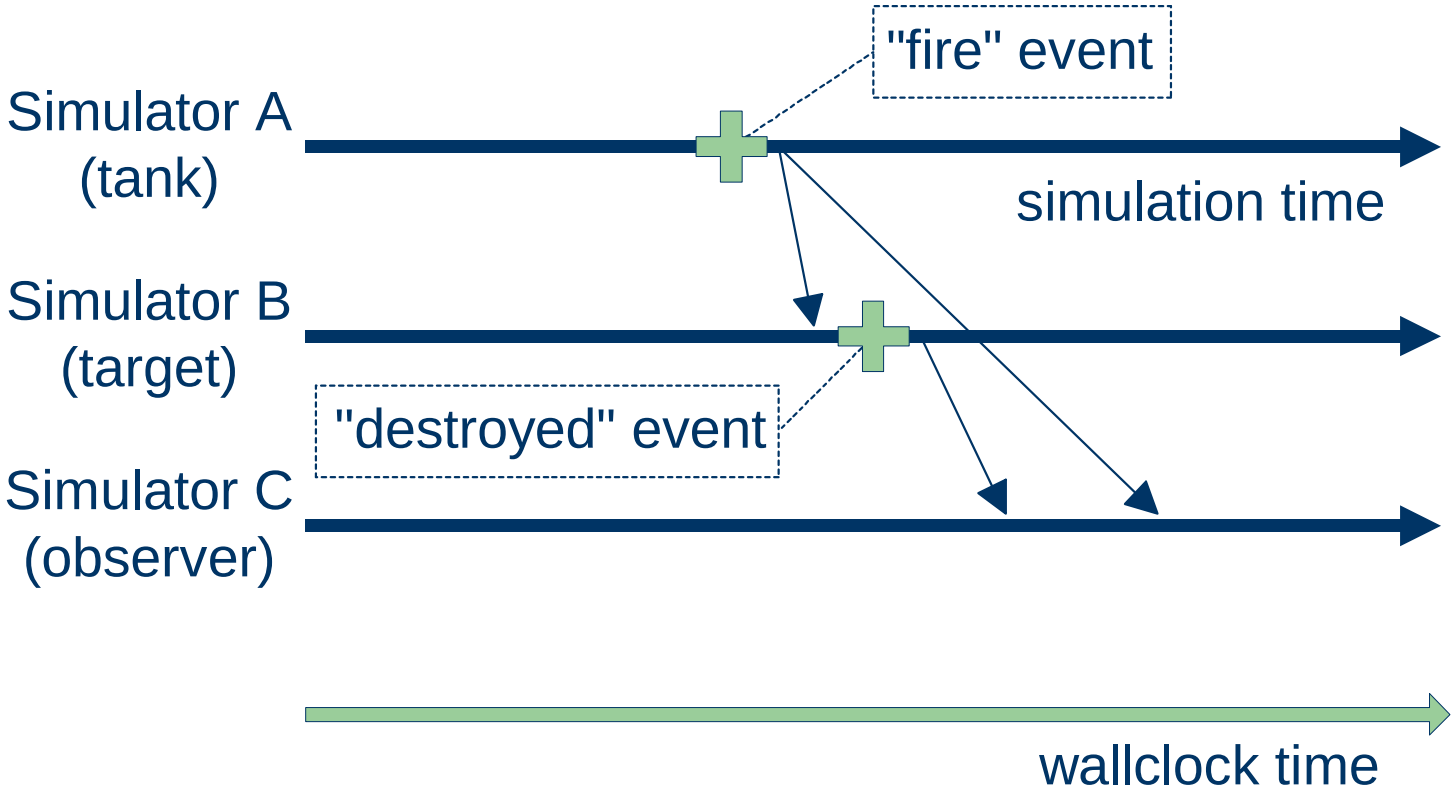
What is Time?

- Physical Time
  - The time in the physical system being modeled
  - Example: in a Pearl Harbour simulation physical time might range from 00h00 through 18h00 on 7 December 1941
- Logical / Simulation Time
  - The simulator's representation of time
  - Example: in the above scenario simulation time may be an IEEE `double` in [0.0, 1080.0]; each unit representing one minute of physical time.
- Wallclock Time
  - The "real world" time during which a simulation is run
  - Example: the above scenario might run in the two hours between 13h47 and 15h47 on 19 March 2003

Why Time Management? (1)

- To ensure that the simulated world “correctly” reproduces temporal aspects of the real world being modeled
- To semantically order cause and effect
- To ensure consistency in the state perceived by federates
- To ensure reproducibility of simulation results

Why Time Management? (2)



Why Time Management? (3)

- Two essential and related problems
  1. Make sure things happen when they are supposed to happen
  2. Make sure thing happen in the order in which they are supposed to happen
- The second problem is solvable if order can be defined by the federates
- The first problem is not solvable in general; the RTI tries to do as well as is possible

Design Overview

- Common Time Management Mechanisms
- Design Rationale
- Assumptions
- Allocation of Responsibilities

Common Time Management Mechanisms

The HLA attempts to unify …

- Event Driven Simulation
- Time Stepped Simulation
- Parallel Discrete Event Simulation
- Real- Time Simulation

Event Driven Simulation

- Federate processes local and external events in time stamp order
- Federate time typically advances to the time stamp of the event currently being processed
- Includes most process oriented simulation formalisms

Time Stepped Simulation

- Each time advance of the federate is of a fixed simulation time duration
- Time does not advance to next time step until all simulation activities for current time step are completed
- Time slicing, activity scanning, etc.

Parallel Discrete Event Simulation

- Federate executes on a multiprocessor system
- Internal processes of the federate synchronized using conservative or optimistic synchronization protocols
  - Conservative – internal logical processes process events in time stamp order
  - Optimistic – internal logical processes process events out of order but can recover if semantic violations occur

Real-Time Simulation

- Derive their current simulation time from the wall-clock time
- These simulations typically require interaction with humans and/or physical devices occur in a timely (i.e., responsive) fashion
- Often do not require event processing in time time stamped order

Design Rationale

- Interoperability and Reuse
  - Accommodate the variety of time management mechanisms commonly in use
  - Support heterogeneous time management mechanisms within a single federation
- Transparency
  - Local time management of a given federate not visible to other federates or to the RTI

Assumptions

- No common, global, clock
- All events are time stamped
- A Federate using logical time may not schedule an event in the past
- Federate are not required to generate events in time stamp order

Allocation of Responsibilities

- Time is jointly managed by RTI and federates
- What functionality belongs in the RTI?
- What functionality belongs in each federate?

Event Driven Simulation

- RTI Responsibilities
  - Time Stamp Order message delivery
  - Grant federates permission to advance time when it can guarantee no pending events have a smaller time stamp
- Federate Responsibilities
  - Merge events delivered by RTI with its internal events
  - Explicitly request and receive permission before advancing to the next internal event.

Time Stepped Simulation

Like Event Driven Simulation …

- RTI Responsibilities
  - Time Stamp Order message delivery
  - Grant federates permission to advance time when it can guarantee no pending events have a smaller time stamp
- Federate Responsibilities
  - Merge events delivered by RTI with its internal events
  - Explicitly request and receive permission before advancing to the time step.

Parallel Discrete Event Simulation

- Federates using conservative synchronization are similar to event driven federates
- Federates using optimistic synchronization may have to recover from execution errors
  - Time Warp
  - Roll back

Real-Time Simulation

A real-time federate is responsible for …

- Clock Synchronization
  - Hardware clocks in different locations must match
- Real-time Scheduling
  - Computations must be complete within deadlines
- Time Compensation
  - Extrapolate values by dead-reckoning

Detailed Design

- Conservative Time Management
  - Message Ordering
  - Look ahead
- Advancing Logical Time
- Types of Federates
- Optimistic Time Management
  - Time Warp

Message Ordering

- Receive Order
- Priority Order
- Time Stamp Order
- Causal Order

Receive Order

- FIFO queue of messages
- Low latency
- Good when causality guarantee is not critical (e.g., real-time)
- Often used by DIS Federates

Priority Order

- Incoming messages go into a priority queue ordered by time stamp
- Lower time stamps have higher priority
- Messages may be delivered out of time stamp order
- A message from the past may be delivered to a federate
- Fairly low latency

Time Stamp Order

- A message will be held until RTI can guarantee that no message having a smaller time stamp will later be received
- No message will be delivered to a federate in its past
- Useful for classical discrete event simulations
- Conflict resolution (ordering of concurrent messages) is deterministic

Causal Order

- A partial order by timestamp
- Related events are delivered in time stamp order
- Conflict resolution (Ordering of concurrent messages) is non-deterministic

Look Ahead (1)

- For Time Stamp Order using conservative synchronization
- A federate promises the RTI that it will predict attribute updates and interactions at least L time units ahead of time
- RTI can use this value to determine when it can safely allow a federate to advance its time
- Facilitates higher simulation performance
- A federate may “retract” a event

Look Ahead (2)

Look Ahead is often derived from

- Physical limitations of federates
- Tolerances to temporal inaccuracies
- Time step increment
- Non-preemptive behaviour
- Pre-computed simulation activities

Look Ahead (3)

- Lower Bound on Time Stamp (LBTS)
  - Earliest time the RTI expects it could possibly deliver a message to a particular federate
  - **LBTS(F)=min(T_i+L_i)** over all federates **i** than can send **F** a time stamp ordered message
  - When **LBTS(F)** exceeds the time advance requested by **F**, the RTI can grant **F** its time advance
- RTI also supports Zero lookahead
  - Time Advance Request Available
  - Next Event Request Available

Optimistic Synchronization (1)

- Allow message processing out of time stamp order but provide rollback recovery mechanism
- Time Stamp Order Delivery …
  - RTI no longer guarantees order message delivery
  - Federate must tell RTI its current logical time
  - RTI provides LBTS to optimistic federates
- RTI provides message queue to federate
  - Flush Queue Request(t)
  - Flush Queue Grant

Time Warp – Optimistic Synchronization (2)

- Federate optimistically receives and processes messages
- Optimistically generated messages sent to other federates (but not necessarily released)
- If execution proves incorrect, federate retracts optimistic messages
- RTI can use LBTS to ensure conservative federates don’t act on optimistic messages
- RTI can use LBTS to ensure optimistic federates don’t receive messages in their past

Type of Federates (1)

Two main characteristics:

- Time Constrained
  - Federate is constrained by the logical time of other federates (i.e., may receive time stamp ordered messages)
- Time Regulating
  - Federate participates in determining the logical time of other federates (i.e., may send time stamp ordered messages)

Type of Federates (2)

Characteristics are independently specified:

- Logical Time Synchronized
  - Both time constrained and regulating
- Externally Time Synchronized
  - Neither time constrained nor regulating
- Logical Time Passive
  - Time constrained but not regulating
- Logical Time Aggressive
  - Time regulating but not constrained

Advancing Logical Time

Time management “cycle”

- Federate invokes RTI requesting time advance
  - Time Advance Request
  - Next Event Request
- RTI delivers $n \geq 0$ messages to federate
  - Reflect attribute value
  - Receive interaction
- RTI grants request
  - Time Advance Grant

Discussion (1)

- Message ordering scheme is selected by the federate developer
  - Different schemes offer different capabilities
  - Often has subtle indirect effects on the semantic correctness of the federation
  - Reuse/Interoperability of federates clearly requires some customization in this regard

Discussion (2)

- RTI does not guarantee timeliness; rather,
  - it helps enforce correctness (as defined by federate declarations to the RTI)
  - it is optimized so as to minimize run-time overhead
- Timeliness and other run-time performance considerations are dependent on the federation design and composition
- Designers want large look-ahead in order increase chance that messages are delivered on time

Implementation Notes

- Client-side library
- Single-threaded reference implementation
  - Like co-operative multitasking system
  - Federate must explicitly yield by calling **`tick()`**