Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 16668

To cite this version: Li, Yanxuan and Cardoso, Janette and Siron, Pierre Adding time-step time management to a distributed Ptolemy-HLAcerti framework. (2015) In: 11th Biennial Ptolemy Miniconference, 16 October 2015 (Berkeley, United States).

Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr
Adding Time-Step Management to Distributed Ptolemy-HLACerti framework

Yanxuan Li, Janette Cardoso, Pierre Siron

11th Biennial Ptolemy Miniconference
Berkeley, October 16, 2015
Plan

• Introduction
• High Level Architecture (HLA-CERTI)
• PTII-HLA Framework
• Example Producer/consumer (NER, TAR)
• Conclusion & Perspectives
Distributed Simulation

WHY?

• System itself is distributed

• System is too complex and/or too much models
  – Reduce the simulation time
  – Enable larger simulations
  – Integrate several (different) simulators into a single simulation environment.
Cyber-Physical System

- Heterogeneous models
- Distributed simulation

Cyber
Science

Control

Physics

HMI

WP

Propeller

Engine

Sensors

IMU

Ultrason

Order

Control

Physical Model

Display

≠ disciplines
HLA High Level Architecture

- High Level Architecture for distributed discrete event simulations
- IEEE standard (1516)
- Interoperability and reuse

Federate = Simulator
Federation = Distributed simulation
<table>
<thead>
<tr>
<th>Areas</th>
<th>Services</th>
<th>Description (non-formal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation</td>
<td>createFederationExecution()</td>
<td>create a federation</td>
</tr>
<tr>
<td></td>
<td>joinFederationExecution()</td>
<td>join a federation</td>
</tr>
<tr>
<td></td>
<td>resignFederationExecution()</td>
<td>quit a federation</td>
</tr>
<tr>
<td></td>
<td>destroyFederationExecution()</td>
<td>destroy a federation</td>
</tr>
<tr>
<td></td>
<td>registerFed...Sync...Point()</td>
<td>register a synchronization point</td>
</tr>
<tr>
<td></td>
<td>sync...PointReg...Succeeded()</td>
<td>register synchro point succeeded</td>
</tr>
<tr>
<td></td>
<td>announceSynchro...Point()</td>
<td>wait a synchronization point</td>
</tr>
<tr>
<td></td>
<td>synchronizationPointAchieved()</td>
<td>release from a synchro, point</td>
</tr>
<tr>
<td></td>
<td>federationSynchronized()</td>
<td>announce synchronization</td>
</tr>
<tr>
<td></td>
<td>tick()</td>
<td>allow to get callbacks from RTI</td>
</tr>
<tr>
<td>Declaration</td>
<td>publishObjectClass()</td>
<td>declare publication of a class</td>
</tr>
<tr>
<td></td>
<td>subscribeObj...ClassAttributes()</td>
<td>subscribe to a class</td>
</tr>
<tr>
<td></td>
<td>unsubscribeObjectClass()</td>
<td>unsubscribe to a class</td>
</tr>
<tr>
<td></td>
<td>unpublishObjectClass()</td>
<td>unpublish a class</td>
</tr>
<tr>
<td>Object</td>
<td>registerObjectInstance()</td>
<td>register an object instance</td>
</tr>
<tr>
<td></td>
<td>discoverObjectInstance()</td>
<td>for object instances discovering</td>
</tr>
<tr>
<td></td>
<td>updateAttributeValues(), UAV</td>
<td>send &amp; update value</td>
</tr>
<tr>
<td></td>
<td>reflectAttributeValues(), RAV</td>
<td>receive updated value</td>
</tr>
<tr>
<td>Time</td>
<td>enableTimeRegulation()</td>
<td>declare federate is regulator</td>
</tr>
<tr>
<td></td>
<td>timeRegulationEnabled()</td>
<td>federate as regulator succeeded</td>
</tr>
<tr>
<td></td>
<td>enableTimeConstrained()</td>
<td>declare federate constrained</td>
</tr>
<tr>
<td></td>
<td>timeConstrainedEnabled()</td>
<td>federate as constrained succeeded</td>
</tr>
<tr>
<td></td>
<td>timeAdvanceRequest(), TAR</td>
<td>ask to advance federate’s time</td>
</tr>
<tr>
<td></td>
<td>timeAdvanceGrant() TAG</td>
<td>notify time advancement granted</td>
</tr>
<tr>
<td></td>
<td>nextEventRequest(), NER</td>
<td>ask to advance federate’s time</td>
</tr>
</tbody>
</table>
HLA Time Management

- Time Advancing mechanisms: the federation will be conservative, deterministic and repeatable.
- The federates can be:
  - Time-Stepped (TAR)
  - Event-Driven (NER)

HLA Object Management

- UAV(object, attribute, timestamp)
- RAV(object, attribute, timestamp)
HLA Time Management

- **Federate**
  - timeAdvanceRequest (TAR)
  - timeAdvanceGrant (TAG)
  - nextEventRequest (NER)

**HLA Object Management**

- **Federate f1**
  - reflectAttributeValues (RAV)
  - updateAttributeValues (UAV)

- **Federate f2**
  - tick()
**Bridges between PTII and HLA standard**

<table>
<thead>
<tr>
<th>PTII</th>
<th>HLA standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Instance</td>
<td>Federate</td>
</tr>
<tr>
<td>Model Time: $t$</td>
<td>Logical Time: $Lt$</td>
</tr>
<tr>
<td>Data (Token)</td>
<td>Data (Attribute of an object class instance)</td>
</tr>
<tr>
<td><strong>Event</strong> $e(t, n, val)$</td>
<td><strong>Event</strong> $e(obj, val, Lt)$</td>
</tr>
<tr>
<td>Director advances time</td>
<td>RTI advances time</td>
</tr>
<tr>
<td>Input &amp; Output ports</td>
<td>HlaPublisher + HlaSubscriber (UAV+RAV)</td>
</tr>
</tbody>
</table>

**Time Management**
- Time advancing in Ptolemy > uses the advancing time services of HLA
- Interface between DE director and RTI: decorator **HlaManager**

**Object Management** - actors for translating:
- A ptII-event from an out-port to a UAV service: new actor **HlaPublisher**
- RAV service from RTI to a ptII-event in an in-port: new actor **HlaSubscriber**
PtII-HLA framework

e(t,n,val)

Run-Time Infrastructure (RTI) : CERTI
PtII-HLA framework

Run-Time Infrastructure (RTI): CERTI

UAV(o1,val,Lt' = g_uav(f_uav(t)))

RAV(o1,val,Lt')

Time lines

Ptolemy

HLA in PtII

HLA

\[ t_{\text{nextPointInTime}_hla} = t_{\text{current}_hla} + T_{s_hla} \]
PtII-HLA framework

Sending

Ptolemy actor \( \text{preUAV}(t_{\text{preUAV}}) \) \( \rightarrow \) HlaPublisher \( \rightarrow \) UAV(\( L_{\text{uav}} \)) \( \rightarrow \) RTI

\[ \text{preUAV}(t_{\text{preUAV}} : \text{Time}) \xrightarrow{f_{\text{UAV}}} \text{pUAV}(t_{\text{uav}} : \text{Time}) \xrightarrow{g_{\text{UAV}}} \text{UAV}(L_{\text{uav}} : \text{CertiLogicalTime}) \]

\( \text{ptolemy-event} \) \( \text{uav-event} \) \( \text{TSO messages} \)

Receiving

RTI \( \text{RAV}(L_{\text{trav}}) \) \( \rightarrow \) HlaSubscriber \( \rightarrow \) Ptolemy actor

\[ \text{RAV}(L_{\text{trav}} : \text{CertiLogicalTime}) \xrightarrow{g_{\text{RAV}}} \text{pRAV}(t_{\text{rav}} : \text{Time}) \xrightarrow{f_{\text{RAV}}} \text{folRAV}(t_{\text{folRAV}} : \text{Time}) \]

\( \text{TSO messages} \) \( \text{rav-event} \) \( \text{ptolemy-event} \)
Events processing

- $e_3(t_3, n_3, val_3, Actor_3)$
- $preUAV_1(t_1, n_1, val_1, HlaPub)$

**CalendarQueue**

**DE Director**
- stopTime: 40.0

**producer1**

**Run-Time Infrastructure (RTI) : CERTI**

$UAV(o_1, val_1, Lt_1 = g_{uav}(f_{uav}(t_1)))$

$preUAV_1(t_1, n_1, val_1)$
Events processing

**HLA in PtII**

\[ pUAV_1(f_{uav}(t_1)), val_1 \]

**CalendarQueue**

\[ e_{3}(t_3, n_3, val_3, Actor_3) \]

**preUAV_1(t_1, n_1, val_1, HlaPub)**

**DE Director**

- **prod**
  - **HLA**
  - **CERTI**
- **stopTime**: 40.0
- **producer1**
  - **lah**: 0.1
  - **Ts**: 10.0

**preUAV_1(t_1, n_1, val_1)**

**UAV(o_1, val_1, Lt_1 = g_{uav}(f_{uav}(t_1)))**

**Run-Time Infrastructure (RTI) : CERTI**

\[ \text{NER} : f_{uav}(t) = t + \text{lah} \]

\[ \text{TAR} : f_{uav}(t) = \begin{cases} t_{\text{current}_\text{hla}} + \text{lah}, & \text{if } t < t_{\text{current}_\text{hla}} + \text{lah} \\ t, & \text{otherwise} \end{cases} \]
Events processing

CalendarQueue

pRAV₁(f_uav(t₁), val₁)

HLA in PtII

DE Director

prod

HLA

CERTI

stopTime: 40.0

lah: 0.1

Ts: 10.0

preUAV₁(t₁, n, val₁)

DE Director

val

HLA

CERTI

stopTime: 41.0

lah: 0.1

Ts: 10.0

UAV(o₁, val₁, Lt₁ = g_uav (f_uav (t₁)))

RAV(o₁, val₁, Lt₁)

Run-Time Infrastructure (RTI) : CERTI
Events processing

Run-Time Infrastructure (RTI): CERTI

\[ \text{pRAV}_1(t, f_{uav}(t), v_l) \]

\[ \text{CalendarQueue} \]

\[ \text{f}_{\text{RAV}}(1) = f_{\text{uav}}(t_1) \]

\[ \text{UAV}(o_1, v_l, L_t = g_{\text{uav}}(f_{\text{uav}}(t_1))) \]

\[ \text{RAV}(o_1, v_l, L_t) \]

\[ \text{preUAV}_1(t_1, n_1, v_l) \]

\[ f_{\text{RAV}}: t_{\text{RAV}} \rightarrow t_{\text{fOLRAV}} \]

\[ NER: f_{\text{RAV}}(t) = t \]

\[ TAR: f_{\text{RAV}}(t) = \begin{cases} t_{\text{nextPinTime}} & \text{if } t \in [t_{\text{current}} \rightarrow t_{\text{nextPinTime}}] \\ t & \text{if } t > t_{\text{nextPointInTime}} \end{cases} \]

\[ \text{f}_{\text{RAV}}(f_{\text{uav}}(t_1), n_1, v_l, \text{HlaSub}) \]

\[ \text{pRAV}_1(t_1, f_{\text{uav}}(t_1), v_l) \]
Data Management with timestamps

- **Time "synchronization"**
  - A local PtII event is safely computed if no (external) HLA events can arrive with a smaller timestamp
  - A PtII federate declares its time advancement proposal to the federation through `proposeTime()`
Distributed Producer/Consumer

\[ e_1(9, 1, 1.0), e_2(15, 1, 2.0), e_3(19, 1, 3.0), e_4(39, 1, 4.0) \]

**Producer**

\[ f_{uav}(t) = t \]

**TAR**

**NER**

\[ f_{uav}(t) = t + lah \]

\[ f_{rav}(t) = t_{nextPinTime_hla} \]

\[ f_{rav}(t) = t \]
Conclusion
• An easy way to produce a HLA federate from a Ptolemy model
• A way to distribute the execution of a Ptolemy model + hardware-in-the-loop
• Time management extend for time-stepped federates (TAR mechanism)

Perspectives
• Compare TAR and NER performance and define related applications
• Implement TARA and NERA mechanisms: TAR and NER with lookahead = 0.
• Make a more complex application: multi-periodic flight controller ROSACE (Research Open-Source Avionics and Control Engineering).
Thank you for your attention.
Distributed Simulation

Ts = 10, lah = 0.1, TAR

Events List

$e_1(9, 1, 1.0), e_2(15, 1, 2.0), e_3(19, 1, 3.0), e_4(39, 1, 4.0)$

Events List

$e_1(10, 1, 0.5)$
**Distributed Simulation**

$Ts = 10, lah= 0.1, TAR$

**Events List**

$e_1(9, 1, 1.0), e_2(15, 1, 2.0), e_3(19, 1, 3.0), e_4(39, 1, 4.0)$

$e_1(10, 1, 0.5), e_2(20, 1, 1.0), e_3(20, 2, 1.5)$
Distributed Simulation

Ts = 10, lah = 0.1, TAR

**Events List**

- \( e_1(9, 1, 1.0) \)
- \( e_2(15, 1, 2.0) \)
- \( e_3(19, 1, 3.0) \)
- \( e_4(39, 1, 4.0) \)

**Events List**

- \( e_1(10, 1, 0.5) \)
- \( e_2(20, 1, 1.0) \)
- \( e_3(20, 2, 1.5) \)
- \( e_4(40, 1, 2.0) \)
## Algorithms of time management

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>HLA</th>
<th>HLA/CERTI</th>
<th>PTII/HLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimistic</td>
<td>X</td>
<td>Non</td>
<td>Non</td>
</tr>
<tr>
<td>Conservative 1&lt;sup&gt;st&lt;/sup&gt; generation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chandy, Misra &amp; Bryan 79</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(lookahead &gt; 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conservative 2&lt;sup&gt;nd&lt;/sup&gt; generation</td>
<td></td>
<td></td>
<td>Other RTI?</td>
</tr>
<tr>
<td>State global computation, Mattern</td>
<td>X</td>
<td>Non</td>
<td>Other RTI?</td>
</tr>
<tr>
<td>Null Prime Message protocol</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Superdense time?

- In accordance with the HLA standard, time representation could be PtII superdense time
  - A requirement?
  - Requires very high C++ skills
- Sometimes, events with the same HLA timestamp are transformed in events with different PtII timestamps
- Actor's ranking
  - Lost in the distribution
  - Problems for an automatic distribution of PtII (work in progress)
Reference

1. R. M. Fujimoto. Time Management in the high level architecture
6. Y. Li. A Distributed Simulation Environment for Cyber-physical systems.
HLA Time Management

- Distributed Events Ordering <-> Time Management <-> Time Advancing mechanisms
- The federation will be conservative, deterministic and repeatable.
- The federates can be:
  - Time-Stepped (TAR)
  - Event-Driven (NER)

HLA Object Management

- UAV(object, attribute, timestamp)
- RAV(object, attribute, timestamp)
Data & Time Management

• Time "synchronization"
  – A local PtII event is safely computed if no (external) HLA events can arrive with a smaller timestamp
  – A PtII federate declares its time advancement proposal to the federation through `proposeTime()`