



High Level Architecture for Distributed Simulations



Oleg Kachirski, Dr. Ratan Guha
University of Central Florida, USA

August 2, 2005

Introduction



- Parallel/Distributed Systems – composed of multiple interconnected computers
- Difference between Parallel & Distributed Systems
- Benefits of P/D Simulation System:
 - Reduced execution time
 - Geographical distribution
 - Integrating multiple computer architectures
 - Fault tolerance
- Usage: high-performance computing (queuing systems), defense community (SIMNET, DIS, HLA), entertainment (VR, gaming, training systems)

Hardware Platforms



- Shared-Memory Multiprocessors
 - Shared variables via [UMA](#) and [NUMA](#)
 - High-speed interconnection network between CPU/cache modules and memory banks, I/O devices
- Distributed-Memory Multicomputers
 - All communication via message passing
 - CPU/cache/memory modules interconnected via network
- SIMD Machines
 - Instructions broadcast by control unit to each processing element (PE)
 - All PEs are network-interconnected

Current Cluster Architecture - 1



- SCEROLA Cluster Multicomputer (shared):
 - 65-node cluster, each node containing:
 - AMD T-Bird 900MHz processor, 256MB RAM (133MHz)
 - 9GB Hard Disk (NFS, emulating shared-disk)
 - Fast Ethernet 100Mbps interconnection network
 - System clock synchronized via message-passing
 - Linux RedHat 7 OS (individual copy on each processor)
 - MPICH 1.2.1
- Issues:
 - Shared system with multiple users
 - May produce biased simulation results

Current Cluster Architecture - 2



- ARIEL Symmetric Multiprocessor Cluster Computer:
 - 32-node cluster, each node containing:
 - Dual P4 1.6 GHz processors, 1GB RAM
 - 40GB Hard Disk (NFS, emulating shared-disk)
 - Gigabit Ethernet interconnection network
 - SUN Solaris OS (individual copy on each processor)

Computer Simulation - Introduction



- Simulation execution:
 - Real-time
 - Scaled real-time
 - As-fast-as-possible
- Time flow mechanism:
 - Continuous
 - Discrete:
 - Time-stepped
 - Event-driven
- Object-oriented and object-based simulations

Parallel and Distributed Simulation




- Synchronization Problem
 - Local Causality Constraint (non-decreasing timestamps)
- Lookahead Principle (null messages):

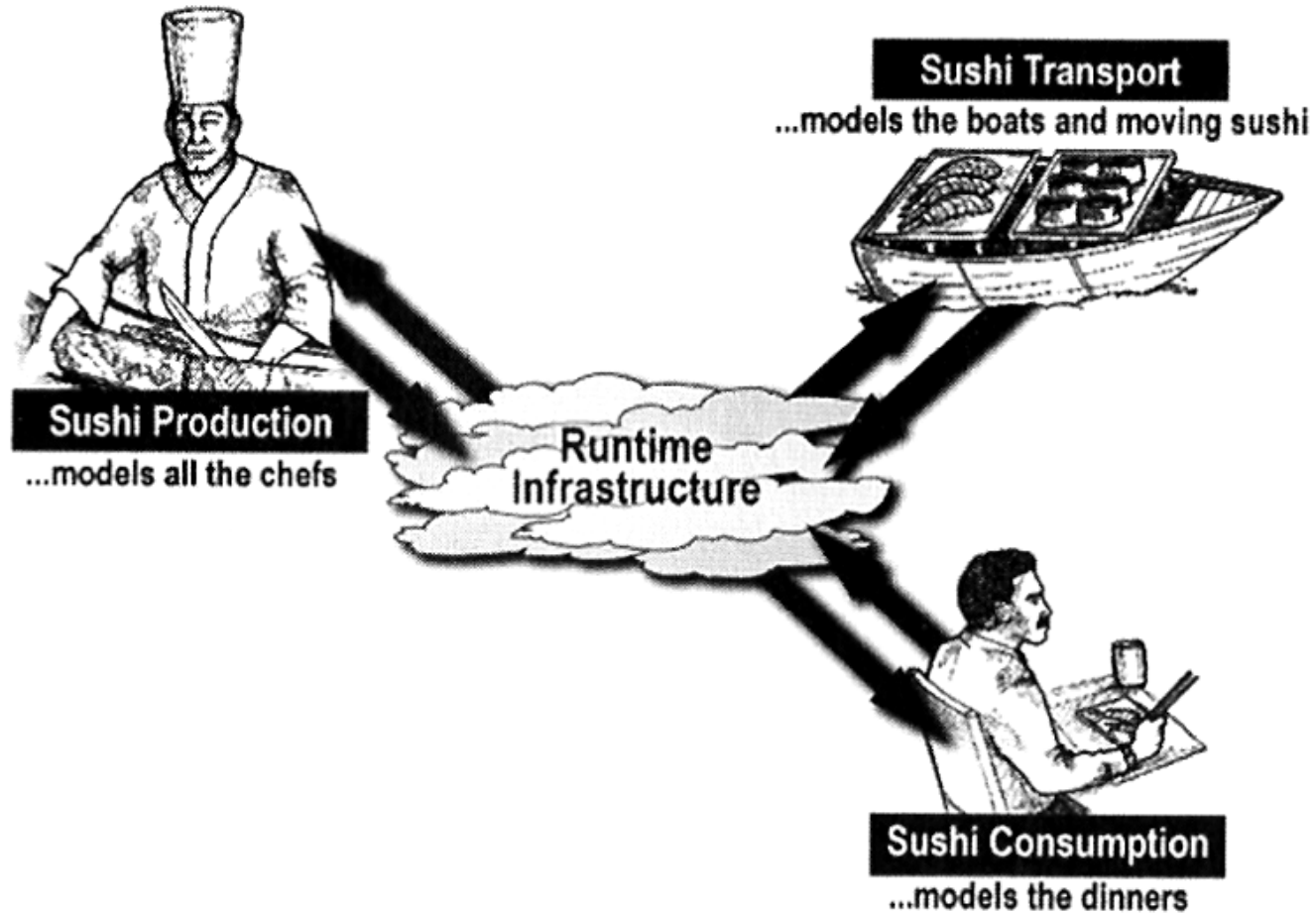


Process event 1, set t_0 to $\text{size}(1)$, then send null message to LP1-3 with timestamp of $\text{size}(2) - \text{size}(1) = \text{clock} + \text{lookahead}$.

HLA Introduction

- 
- HLA - High Level Architecture
 - A framework for distributed systems implementation
 - A software architecture for large scale distributed simulations
 - Adopted by DoD for all modeling and simulation activities

Example – Sushi Restaurant Federation



Terminology



- HLA
- Federation
- Federate
- Federation Execution
- RTI


Federation




Consists of:

- Runtime Infrastructure (RTI)
- Federation Object Model (FOM)
- Federates


Federate

- 
- Member of the federation
 - May consist of a single process or several processes
 - Granularity depends on the implementation
 - Has strict guidelines for defining data and methods

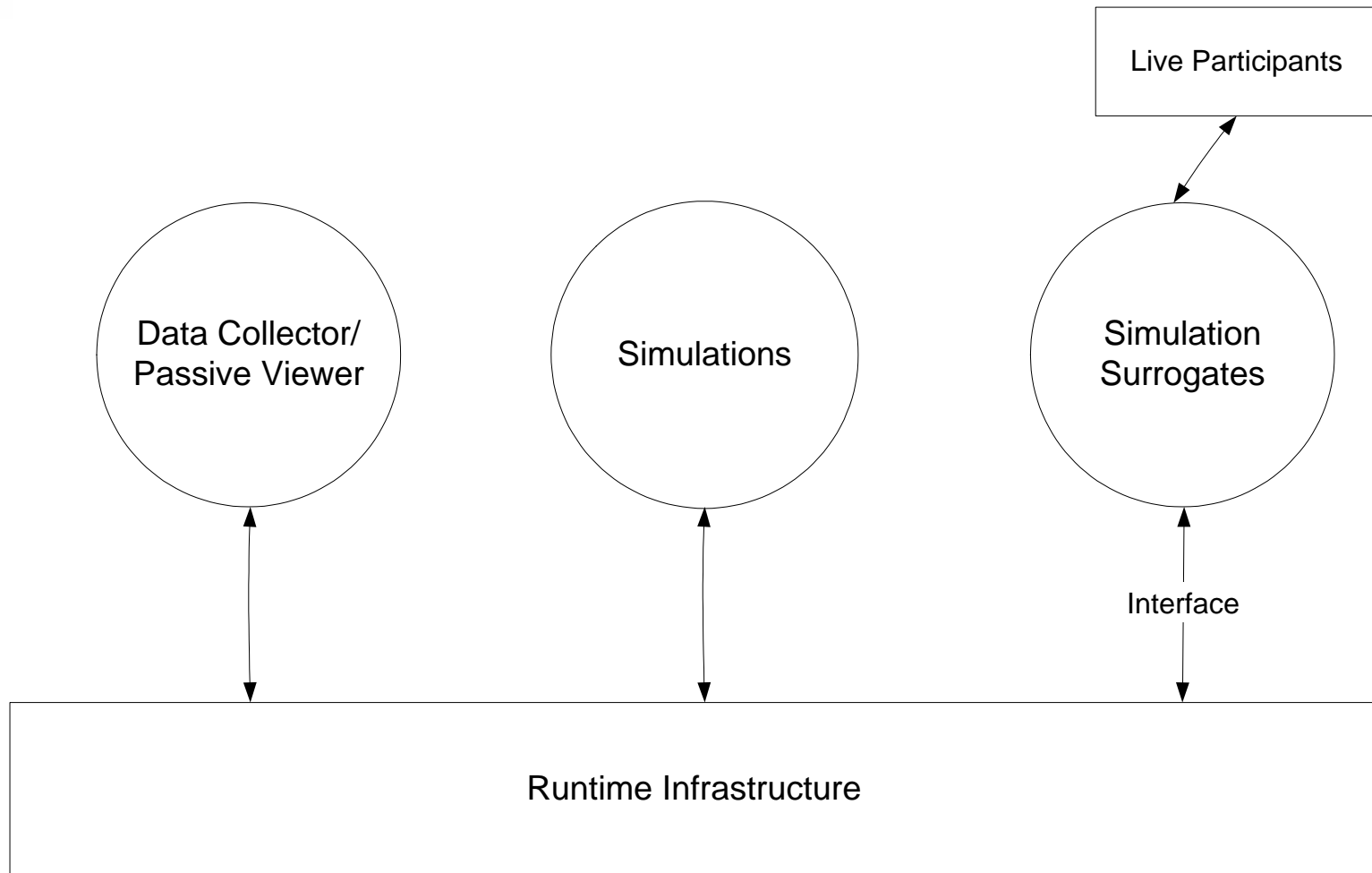
Federation Object Model (FOM)

- 
- Data component of HLA
 - Contains the data types
 - Specifies data type rules
 - Helps a structured communication among federates
 - Furnished to the RTI during federation execution


HLA as Software Architecture

- 
- HLA is a layered architecture
 - Distributed nature is hidden from federates
 - HLA is a data abstraction architecture
 - All federates inherit the same interface
 - All communication is done via RTI
 - HLA is an event-based architecture


Software Components of HLA



Information Model

- 
- Object Model Template (OMT) prescribes the structures for all FOMs
 - FOM is the *vocabulary of data* exchanged through the RTI
 - FOM describes only *shared* data with other federates
 - FOM is a parameter to the RTI at the beginning of an execution
 - RTI does not change when FOM is changed

Federation Rules

- 
- Federations shall have an HLA FOM in accordance with HLA OMT
 - In a federation, all simulation-associated object instances are in the federates
 - All exchanges of FOM data among federates occur via RTI
 - Federates shall interact with the RTI in accordance with the HLA Interface Specification
 - An instance attribute shall be owned by at most one federate at any time

Federate Rules



- All federates have an HLA Simulation Object Model (SOM)
- Federates shall be able to update attributes and manage interactions, as specified in their SOMs
- Federates shall be able to transfer ownership of attributes dynamically, as specified in their SOMs
- Federates shall be able to manage local time

HLA Services



- Federation Management
 - by defining a federation execution and membership
 - by accomplishing federation-wide operations
- Declaration Management
 - Federates declare their intent to produce/publish and consume/subscribe to data
- Object Management
 - Send and receive interactions
 - Register new instances of an object class and update its attributes

HLA Services (Continued)



- Ownership Management
 - This service in the RTI implements the HLA's notion of security while simulating an entity
- Time Management
 - Allow each federate to advance its logical time in coordination with other federates
 - Control the synchronized delivery of time-stamped events
- Data Distribution Management
 - Control the producer-consumer relationships among federates
 - Manages relationships in terms of object instances and routing spaces

Demo 1 – Chat System



- Objective: to develop personal network communication system using HLA
- Tasks:
 - Work in groups of 2 people (1 per workstation)
 - Start RTI on one workstation (run rti2.bat file in HLA folder)
 - Start Chat federate on each workstation from Examples\Chat folder:

```
java -classpath prti.jar;jgl3.1.0.jar;"%classpath%" Chat <IP Address>
```
 - Type any text in the window at the prompt
- Result: Each workstation sends data via RTI communication bus, and results are displayed at another workstation.
 - Note: Corresponding IP address will be displayed at each workstation

Demo 1 - Code Details



Java file: Chat.java

1. Try to create a federation. If it exists already, then join it:

```
_rtiAmbassador = RTI.getRTIAmbassador(rtiHost, 8989);
    try {
        _rtiAmbassador.createFederationExecution(
            "ChatRoom", new URL("file:chat.fed"));
    }
catch (FederationExecutionAlreadyExists ignored) { }
_rtiAmbassador.joinFederationExecution("Chat", "ChatRoom", this);
```

Demo 1 - Continued



2. Create point of interaction from FOM specifications:

```
_messageId = _rtiAmbassador.getInteractionClassHandle(
    "Communication");
_parameterIdText = _rtiAmbassador.getParameterHandle(
    "Message", _messageId);

_rtiAmbassador.subscribeInteractionClass(_messageId);

_rtiAmbassador.publishInteractionClass(_messageId);
```

Demo 1 - Continued



3. Create a parameter for communication and send an HLA interaction via federate's RTI Ambassador:

```
SuppliedParameters parameters =  
    RTI.suppliedParametersFactory().create(1);
```

```
byte[] value = cmdline.getBytes();
```

```
parameters.add(_parameterIdText, value);
```

```
_rtiAmbassador.sendInteraction(_messageId, parameters,  
    null);
```

Demo 2 – Sorting Server



- Objective: to develop a client - server system using HLA
- Tasks: <http://www.cs.ucf.edu/courses/cda4932.spr02/assign2.html>
 - Work in groups of 2 people (1 per workstation)
 - Start RTI on one workstation (run rti2.bat file in HLA folder)
 - Start Sorting Server federate on one workstation from Examples\Sort folder:
`java -classpath prti.jar;jgl3.1.0.jar;"%classpath%" Server <IP Address>`
 - Start Client federate on another workstation from Examples\Sort folder:
`java -classpath prti.jar;jgl3.1.0.jar;"%classpath%" Client <IP Address>`
 - Repeat running client for a few iterations
- Result: Client workstation sends unsorted data to a Server via RTI, and the Server returns sorted data to the client.

Demo 3 – Distributed Database System



- Objective: to develop a distributed database query system using HLA
- Tasks: <http://www.cs.ucf.edu/courses/cda4932.spr02/assign3.html>
 - Work individually (1 or 2 people per workstation)
 - Open 4 DOS windows
 - Start RTI on a workstation (run rti2.bat file in HLA folder)
 - Start Sorting Server federate 3 on a workstation from Examples\Query folder, one command per each DOS window:

```
java -classpath prt.jar;jgl3.1.0.jar;"%classpath%" Server 1
java -classpath prt.jar;jgl3.1.0.jar;"%classpath%" Server 2
java -classpath prt.jar;jgl3.1.0.jar;"%classpath%" Server 3
```

Demo 3 (cont.-d)



- Start Client federate on a workstation from Examples\Query folder:
`java -classpath prt.jar;jgl3.1.0.jar;"%classpath%" Client`
- Issue a query in the format (any of the following):
`age=24`
`fname=Wendy`
`lname=Arthurs`
- Server will return the search results
- Repeat running client for a few iterations
- Result: Client workstation sends a query to each Database Server via RTI, and each Server returns results of the query to the Client. Each server processes user query (database search operation) in parallel.

References



- “Creating Computer Simulation Systems: an Introduction to the High Level Architecture”, Frederick Kuhl, Richard Weatherly, Judith Dahmann, Prentice Hall PTR. ISBN 0-13-022511-8
- Book website: http://www.mitre.org/tech/hla_book/
- DMSO Software Distribution Center: <https://sdc.dmsso.mil/>
- Pitch pRTI (free implementation 1.3):
<http://www.pitch.se/>
- Oleg Kachirski – HLA website:
<http://www-ece2.engr.ucf.edu/~oka/eel6885/>
- CDA4932 Distributed Systems course website:
<http://www.cs.ucf.edu/courses/cda4932.spr02/index.html>