Distributed Simulation
and
Inherently Distributed Systems

Giuseppe Iazeolla, Andrea D’Ambrogio, Alessandra Pieroni
University of Rome “TorVergata”
Daniele Gianni
ESA ESTEC

Presented by Alessandra Pieroni
Context
Simulation-driven design of systems

Execution Platform

Simulator

System to be designed
This Presentation

Context application to
Inherently Distributed System
(IDS)
Definitions

Inherently Distributed Systems (IDS)

systems that are distributed by their own nature

their subsystems are physically

and

geographically separated
Example IDS’s

- distributed computer systems
  - $m$ geographically separated hosts

- wireless systems
  - (e.g. WiFi/WiMax (IEEE 802.11/16))
  - 1 base-station for each $m$ subscriber-stations
  - $k$ terminal equipments for each SS

- satellite constellations
  - $m$ spatially separated orbiting satellites,
  - 1 ground segment and 1 user segment
Known paradigms for IDS simulation

- **Type-1**: Local Simulation *(LS)*
simulator run by a single host

- **Type-2**: Distributed Simulation *(DS)*
simulator run by a number of hosts
to achieve

  *scalability, aggregation, reusability and parallelism*
Type-3 paradigms for DS

- naturally Distributed Simulation
  \((nDS)\) locates the federates in the same geographic positions of the IDS subsystems

- naturally Distributed Simulation in-the-loop
  \((nDS-il)\) as nDS but one federate left in its natural form
Type-3 paradigms for DS
(obtain Scalability, Parallelism and Representativeness)

$m$ HOSTs
(where to locate them ?)

Simulator is partitioned
into $m$ federates

System to design is **IDS** (with $m$ subsystems)
Example application of 4 paradigms to the simulation driven design of wireless systems
The LS paradigm for Wireless systems

(m+2 subsystems run by 1 single simulation platform)
The DS paradigm for Wireless systems

(m+2 subsystems run by m+2 simulation platforms)

platform locations are anywhere

(no relationship with the subsystems' locations)
The *nDS* paradigm for Wireless systems

The *nDS* paradigm for Wireless systems

the m+2 simulation platforms are located in the same geographic locations of the m+2 subsystems

DSIMday Giornata di studio MIMOS sulla Simulazione Distribuita – 11 Marzo 2011
nDS-il paradigm for Wireless systems

as nDS for the first m+1 subsystems
while the (m+2)th subsystem is left in its real form
Example application of 4 paradigms to the simulation driven design of Satellite Systems
The LS paradigm for Satellite Systems

(m+2 subsystems run by 1 single simulation platform)
The DS paradigm for Satellite Systems

(m+2 subsystems run by m+2 simulation platform) platform locations are anywhere (no relationship with the subsystems locations)

DSIMday Giornata di studio MIMOS sulla Simulazione Distribuita – 11 Marzo 2011
The nDS paradigm for Satellite Systems

Satellite Constellation

Sat1 Services Simulator (Platform 1)
Sat2 Services Simulator (Platform 2)
... Sat m Services Simulator (Platform m)

Ground Segment Simulator (Platform m+1)

Development Team

User Segment

the m+2 simulation platforms are located in the same geographic locations of the m+2 subsystems
The nDS-il paradigm for Satellite Systems

as $nDS$ for the first $m+1$ subsystems

while the $(m+2)$th subsystem is left in its real form
New Software technologies for the nDS and nDS-il paradigms

- an environment (*nDSEnv*)
- a language (*nDSLang*)

Together give a complete software suite that ease the development of *nDS* and *nDS-il* systems.
Using the *nDSEnv* and *nDSL*ang

- You may develop a **DS** systems with **no knowledge** of HLA
  - First develop the **LS** system
  - By use of a **mechanical** process then produce the equivalent **DS** system
- The process generates only a very limited additional amount of LOCs
- The production of the **DS** system is practically **effortless**
Example use of $\text{nDSEnv}$ and $\text{nDSLang}$

$m$-node communication system

Connection matrix

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$\ldots$</th>
<th>$D_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$p_{1,1}$</td>
<td>$p_{1,2}$</td>
<td>$p_{1,3}$</td>
<td>$\ldots$</td>
<td>$p_{1,m}$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$p_{2,1}$</td>
<td>$p_{2,2}$</td>
<td>$p_{2,3}$</td>
<td>$\ldots$</td>
<td>$p_{2,m}$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$p_{3,1}$</td>
<td>$p_{3,2}$</td>
<td>$p_{3,3}$</td>
<td>$\ldots$</td>
<td>$p_{3,m}$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$S_m$</td>
<td>$p_{m,1}$</td>
<td>$p_{m,2}$</td>
<td>$p_{m,3}$</td>
<td>$\ldots$</td>
<td>$p_{m,m}$</td>
</tr>
</tbody>
</table>

($S =$ sending node, $D =$ destination node)
Example use of \texttt{nDSEnv} and \texttt{nDSLlang}

\textbf{LS} version

- one standard single-platform simulator
  - define the \textit{m nodes}
  - define the \textit{mxm links}
  - run the experiments
Transformation from $\mathcal{LS}$ into $\mathcal{DS}$

$\mathcal{DS}$ version (assume 2 Federates)

federate1: nodes 1 and 2
federate2: remaining $m-2$ nodes
Transformation from LS into DS

The DS Federate 1 consists of nodes $S_1$ and $S_2$:

- Such nodes are local to federate 1 and their declaration follows the same statements of the LS system, that can therefore be reused in the DS code.

The DS Federate 2 consists of nodes $S_3$ through $S_m$:

- Such nodes are local to federate 2 and their declaration follows the same statements of the LS system, that can therefore be reused in the DS code.
Transformation from LS into DS

- only need to modify are those link declarations that cross the border between the two sub-matrices, i.e.:

- links from $S_1$ to $D_3, D_4, D_5, ..., D_m$
- links to $S_1$ from $D_3, D_4, D_5, ..., D_m$
- links from $S_2$ to $D_3, D_4, D_5, ..., D_m$
- links to $S_2$ from $D_3, D_4, D_5, ..., D_m$
Transformation from \textit{LS} into \textit{DS}

- The \textit{DS} version of the system is easily derived from the original \textit{LS} version (and also easily automated)
- The largest part of the \textit{DS} code is reused from the \textit{LS} code
- The only statements to modify are the interface declarations between federates
Research open problems
(nDS-il)

1) Reproducibility:
   • The real federate introduces simulation-external phenomena that cannot be reproduced
   • This may make the “reproducibility” of simulation experiments problematic

2) Interface to the real federate:
   • The federate that simulates e.g. the SS station of a wireless system, needs to drive a physical antenna to send packets to the real interconnection infrastructure
   • This requires creating an interface between the SS federate and its antenna system
3) **Relationship problems between the federates network (FN) and the real network (RN)**

- **FN** is the network used by the federates to exchange synchronization and communication messages (could be a dedicated WAN, a public WAN or the Internet itself).
- **RN** is the network part of the simulated system.
- The **FN** delays should be compatible with the **RN** time scale....
Example relationship between federates network \((FN)\) and the real network \((RN)\)
Conclusions

- The DS approach to inherently distributed systems yields simulation scalability, aggregation, reusability and parallelism.

- The nDS and nDS-il approaches yield the additional feature of simulation representativeness.
Conclusions

- The existing distributed simulation tools (such as HLA) may represent an obstacle to the wide adoption of the paradigms.

- A HLA-transparent simulation environment and a simulation language ($nDSEnv$ and $nDSLang$) have been introduced.

- Such technologies overcome the HLA difficulties and allow developing an $nDS$ or $nDS-il$ system as it was a conventional $LS$ system.
Conclusions

- \texttt{nDSEnv} and \texttt{nDSLang} give facilities rarely found in existing distributed simulation technologies:
  - a Java-based language is used
  - the skills needed to develop a \texttt{nDS} or \texttt{nDS-\textit{il}} simulation system are brought down to the standard skills of a \texttt{LS} one
  - once an \texttt{LS} system is obtained, bringing it into \texttt{nDS} or \texttt{nDS-\textit{il}} form can be done with practically no extra effort and without any HLA skill
References


ACK

Work partially supported by funds from the FIRB project on “Software frameworks and technologies for distributed simulation”, from the FIRB project on “Performance evaluation of complex systems”, from the University of Rome Tor Vergata research on “Performance modeling of service-oriented architectures” and from the CERTIA Research Center.
Thank you for your kind attention...