IC2D: Interactive Control and Debug of Distribution

Françoise Baude,
with
Denis Caromel,
Fabrice Huet, Julien Vayssiere,
Alexandre Bergel, Olivier Nano

OASIS Team
INRIA -- CNRS - I3S -- University of Nice Sophia-Antipolis

www.inria.fr/oasis/ProActive

Plan of the talk

- Introduction
- 1) Overview of distributed object-oriented programming and ProActive
- 2) IC2D
  - Features:
    - Visualisation
    - Control
  - Design
- 3) Related work and Conclusion

Introduction

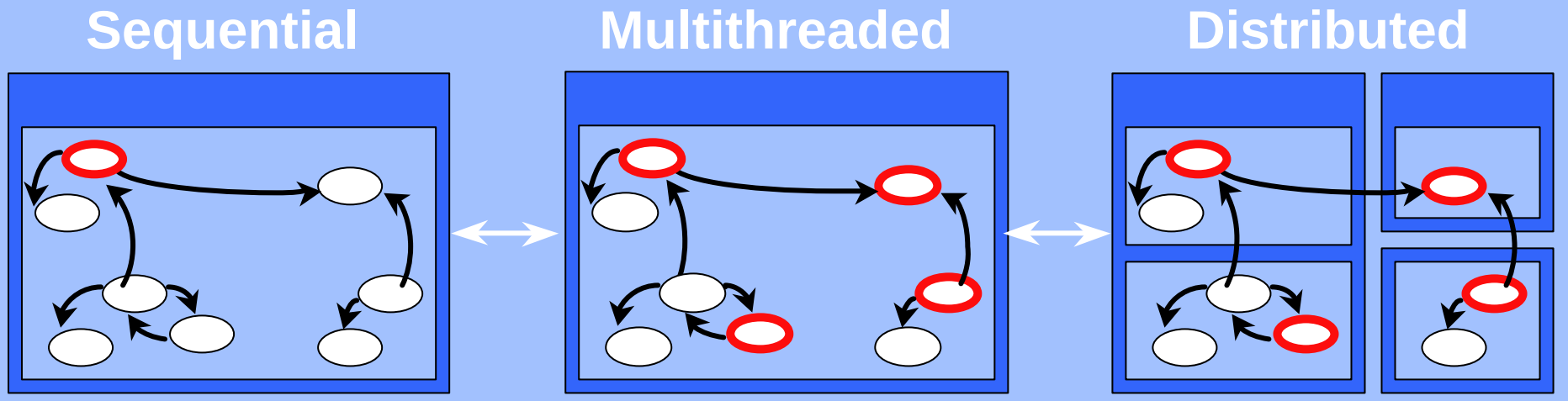
General Context of our research:

- **Parallel and Distributed Object Oriented Programming with two main objectives:**
  - **sequential code reuse**
  - **target any platform (parallel machine, network of local workstations and servers, clusters, grids, ...) without code modification**
- **library: ProActive PDC (previous experiments: Eiffel //, C++ //)**

==> Need of a uniform Graphical User Interface in order to launch, monitor, steer, debug , etc, such distributed, maybe wide and large-scale, computations.

1) Object-Oriented Distributed Programming with ProActive

Objective : Seamless

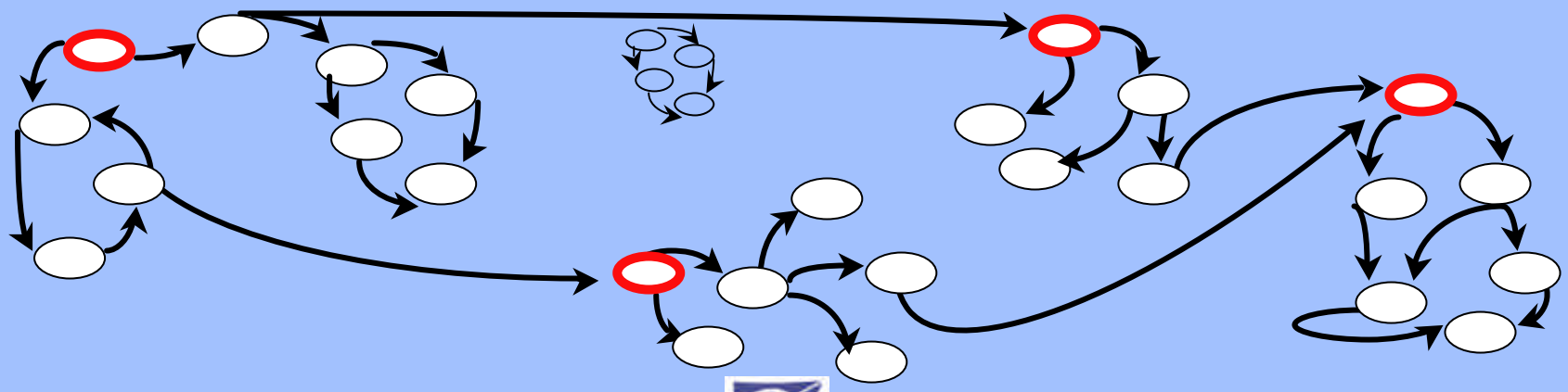


- Most of the time, activities and distribution are not known at the beginning, and change over time
- Seamless implies reuse, smooth and incremental transitions

Reification through a library !

ProActive : model

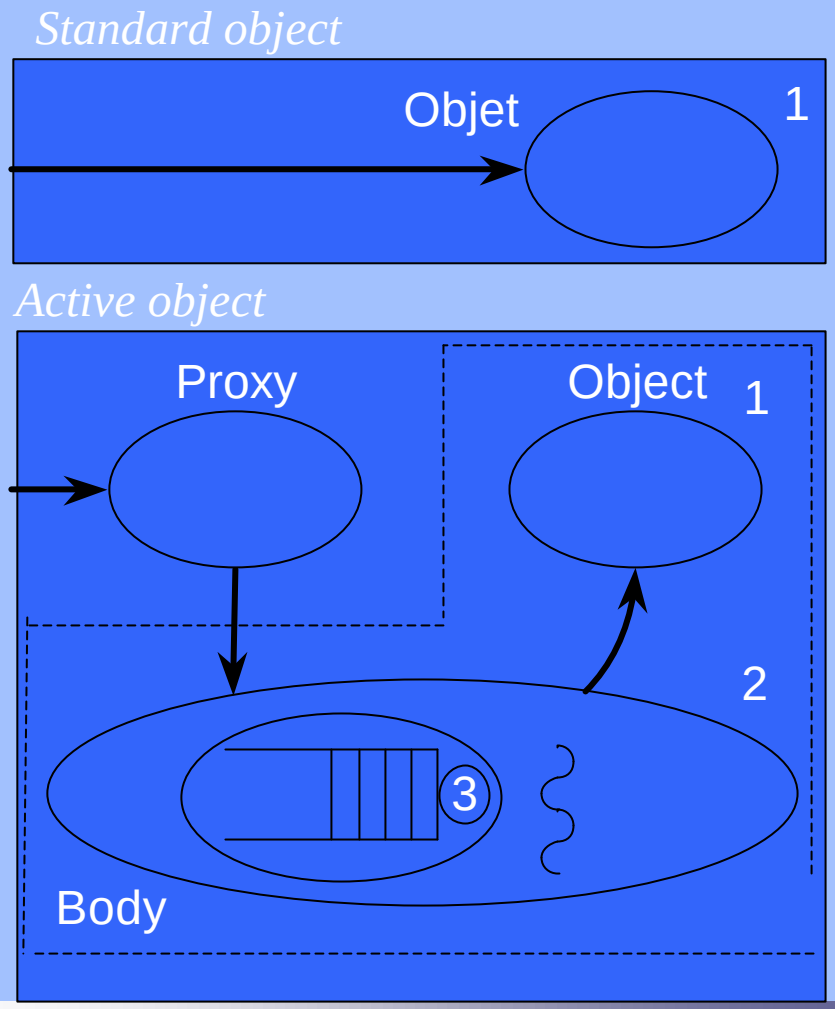
- Active objects : coarse-grained structuring entities (subsystems)
- Each active object: - possibly owns many passive objects
  - has exactly one thread.
- No shared passive objects -- Parameters are passed by deep-copy
- Asynchronous Communication between active objects
- Future objects and wait-by-necessity.
- Full control to serve incoming requests (by reification)



ProActive : Active object

An active object is composed of several objects :

- The object itself (1)
- The body: handles synchronization and the service of requests (2)
- The queue of pending requests (3)



ProActive : Creating active objects

Single inheritance: using interface

An object created with `A a = new A (obj, 7);`

can be turned into an active object:

- **Class-based**

```
class pA extends A implements Active {}
A a = (A)ProActive.newActive(«pA»,params, NODE);
```

The most general case. Allows to provide a specific behavior.

- **Instanciation-based**

```
A a = (A)ProActive.newActive(«A»,params, NODE);
```

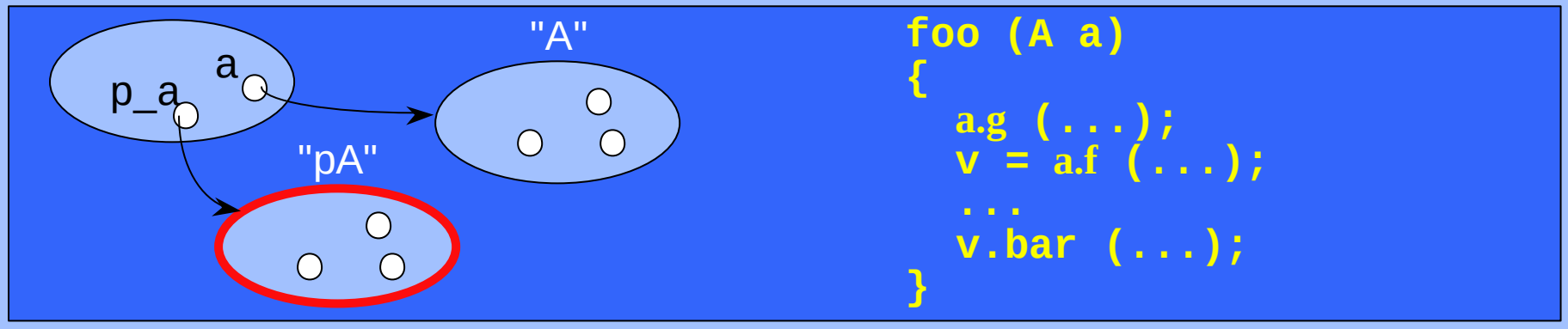
- **Object-based**

```
A a = new A (obj, 7);
a = (A)ProActive.turnActive (a, NODE);
```

ProActive : Reuse and seamless

Two key features:

- Polymorphism between standard and active objects
  - Type compatibility for classes (and not only interfaces)
  - Needed and done for the future objects also
  - Dynamic mechanism (dynamically achieved if needed)



- Wait-by-necessity: inter-object synchronization
  - Systematic, implicit and transparent futures

    Ease the programming of synchronizations, and the reuse of routines

ProActive : Migration of active objects

Migration is initiated by the active object itself through a primitive: migrateTo

Can be initiated from outside through any public method

The active object migrates with:

- all pending requests
- all its passive objects
- all its future objects

Automatic and transparent forwarding of:

- requests (remote references remain valid)
- replies (its previous queries will be fullfilled)

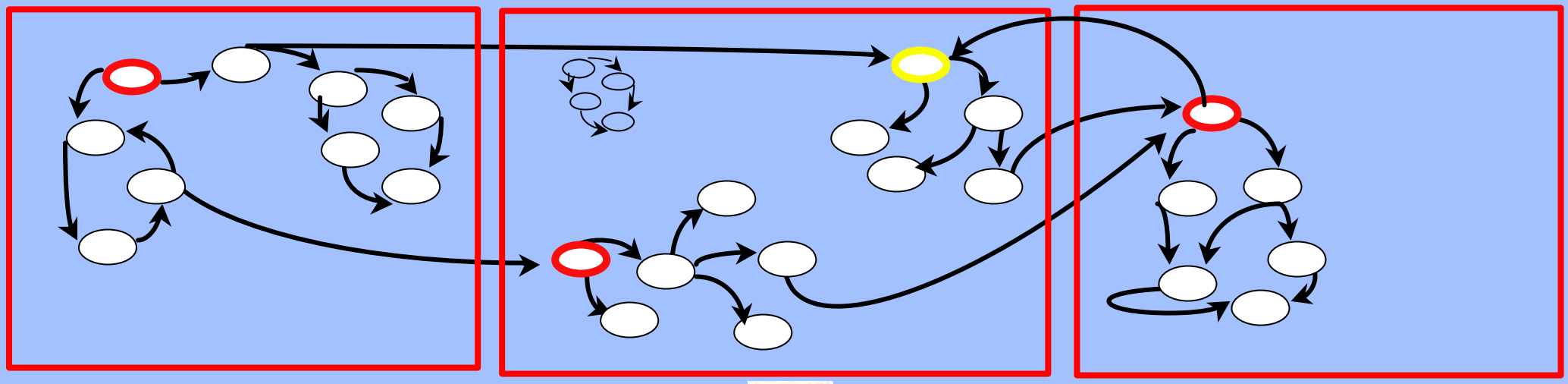
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



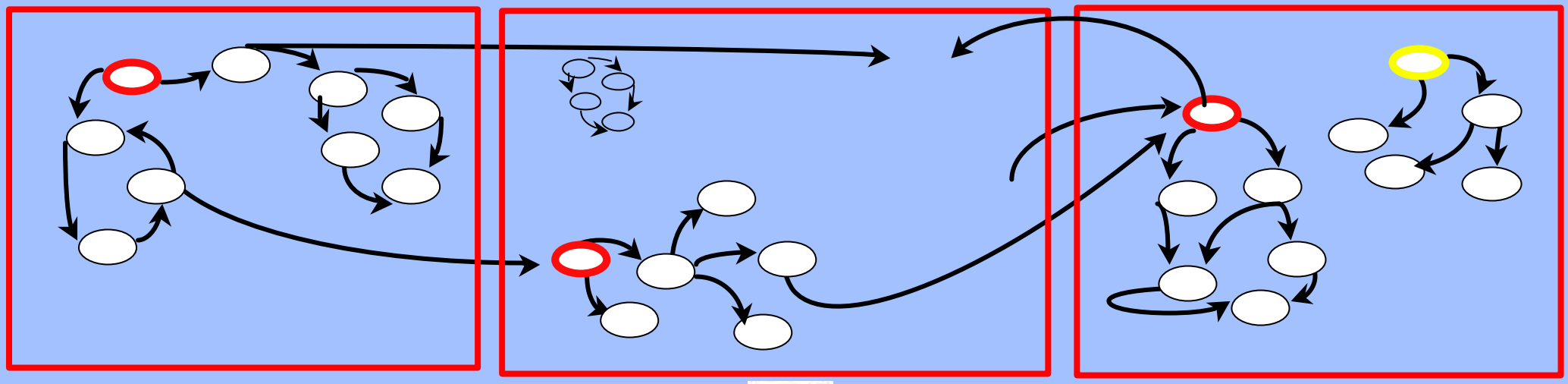
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



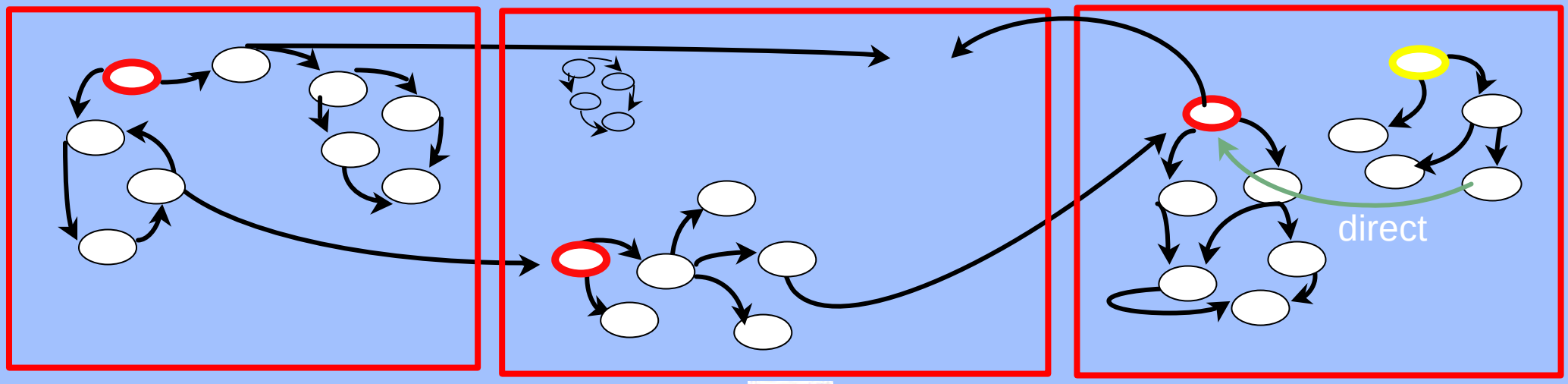
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



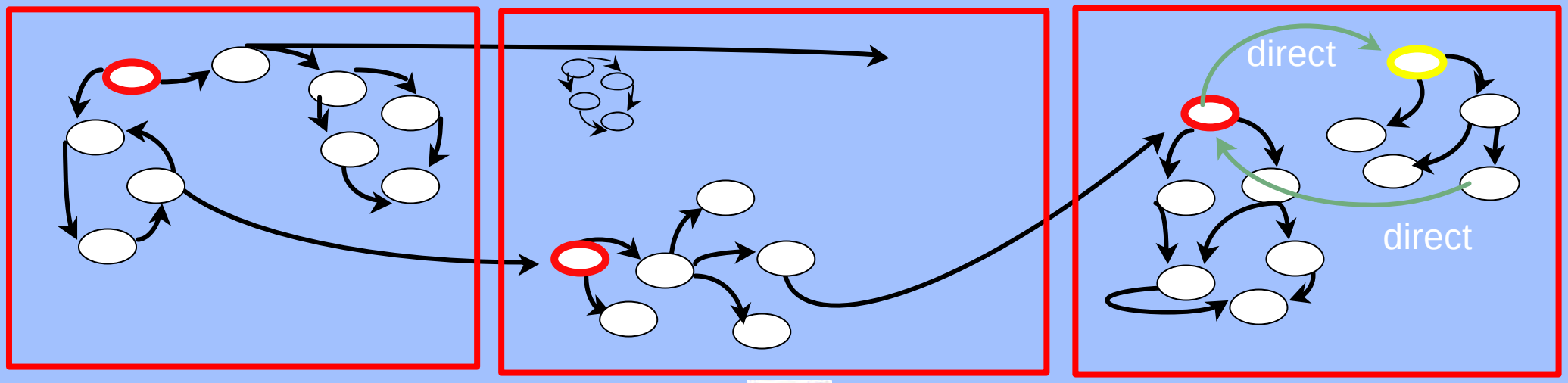
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



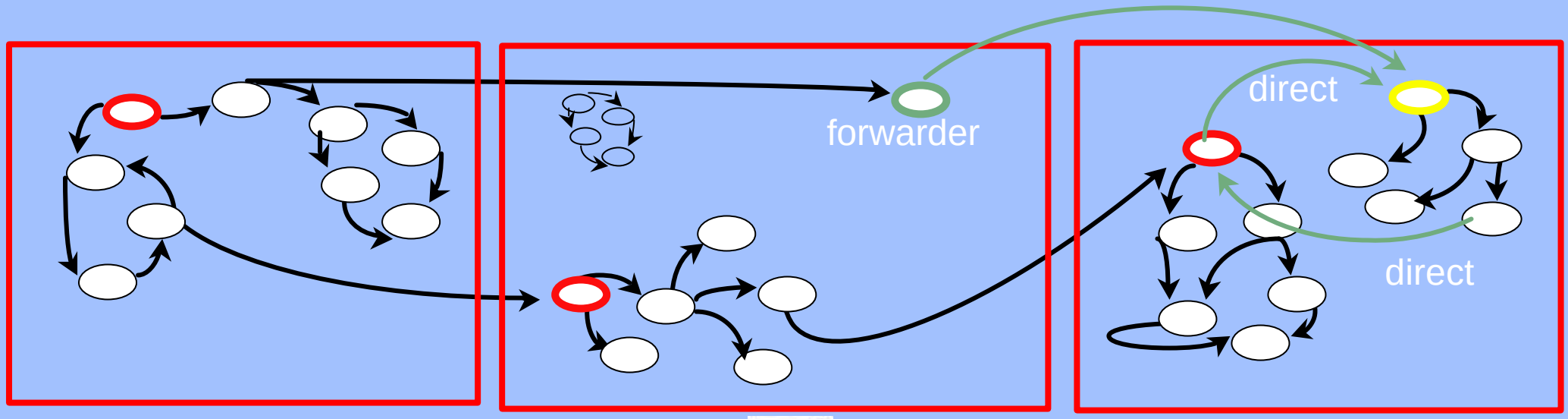
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



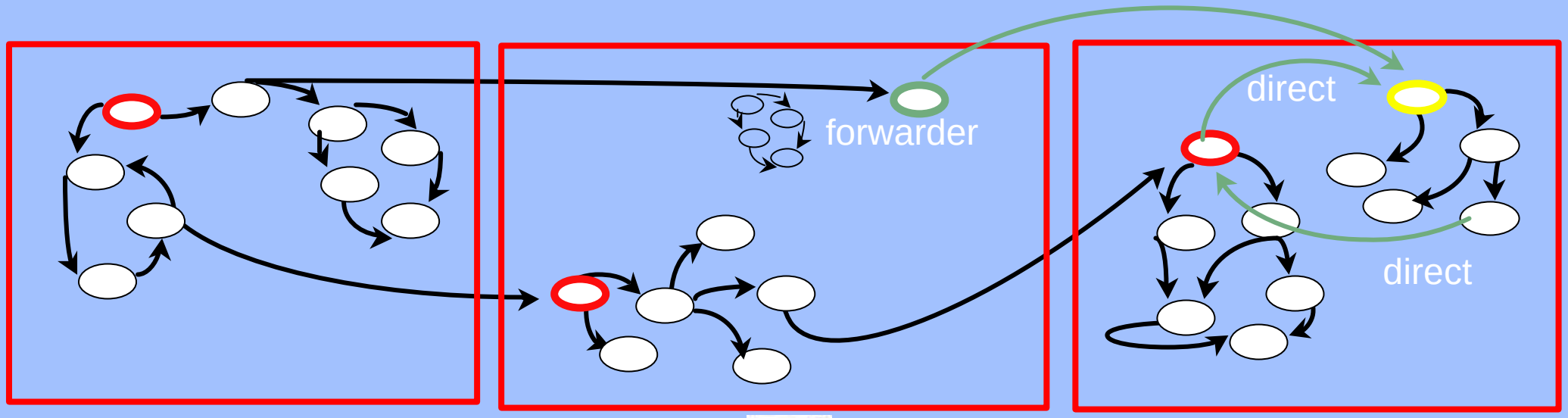
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



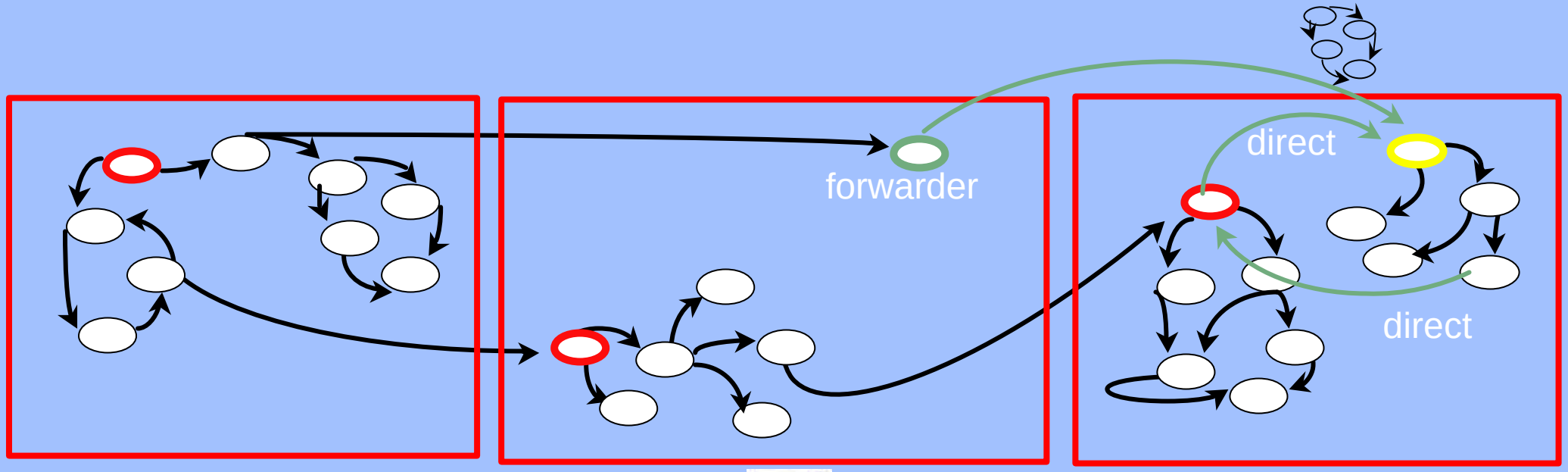
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



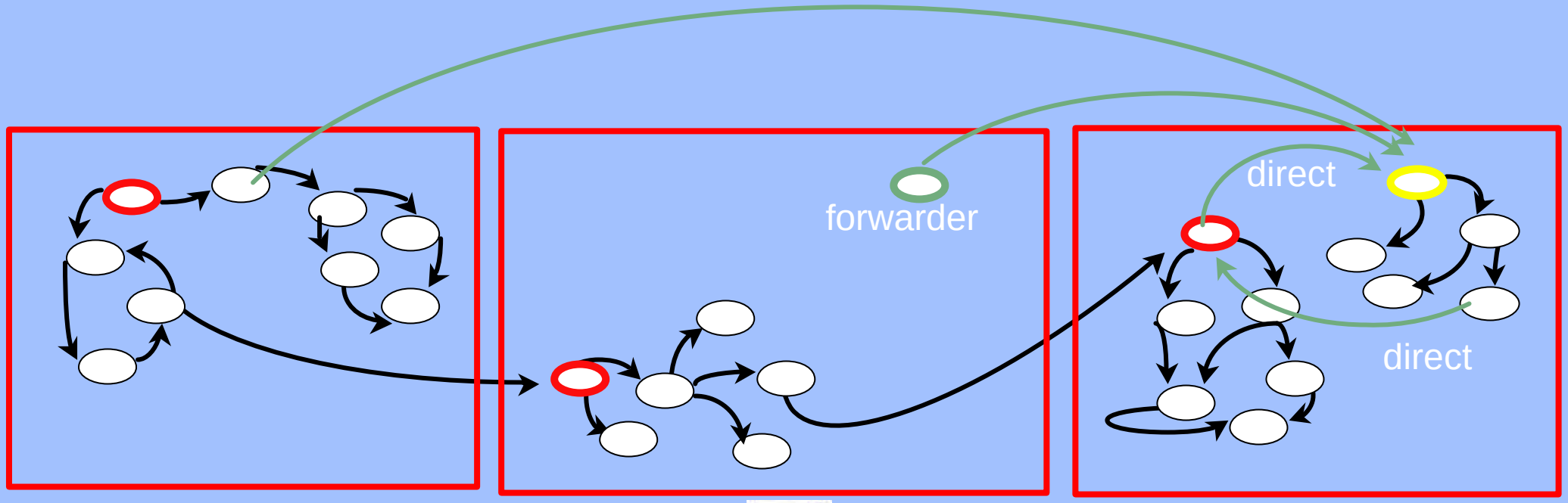
Characteristics and optimizations

Same semantics guaranteed (RDV, FIFO order point to point, asynchronous)

Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



ProActive: summary of Non Functional Properties

Currently in ProActive:

- **Remotely accessible Objects**
  (Classes, not only Interfaces, Dynamic)
- **Asynchronous Communications, Futures**
- **Migration**
- **Visualization and monitoring (IC2D)**

Others:

- **Security (worked on)**
- **Group Communications (worked on)**
- **Communications with disconnected mode**

2) IC2D: Interactive Control and Debug of Distribution

Objectives:

- **Help the deployment of applications**
  - **on any kind of heterogeneous platforms, including the grid**
- **Help the debugging and monitoring of applications**
  - **because there are distributed**

  In a sense, similar to xpvm, xpm2, but applied to ProActive
- **From the monitor -- running if needed as an applet in a browser--, be able to interactively steer the applications**

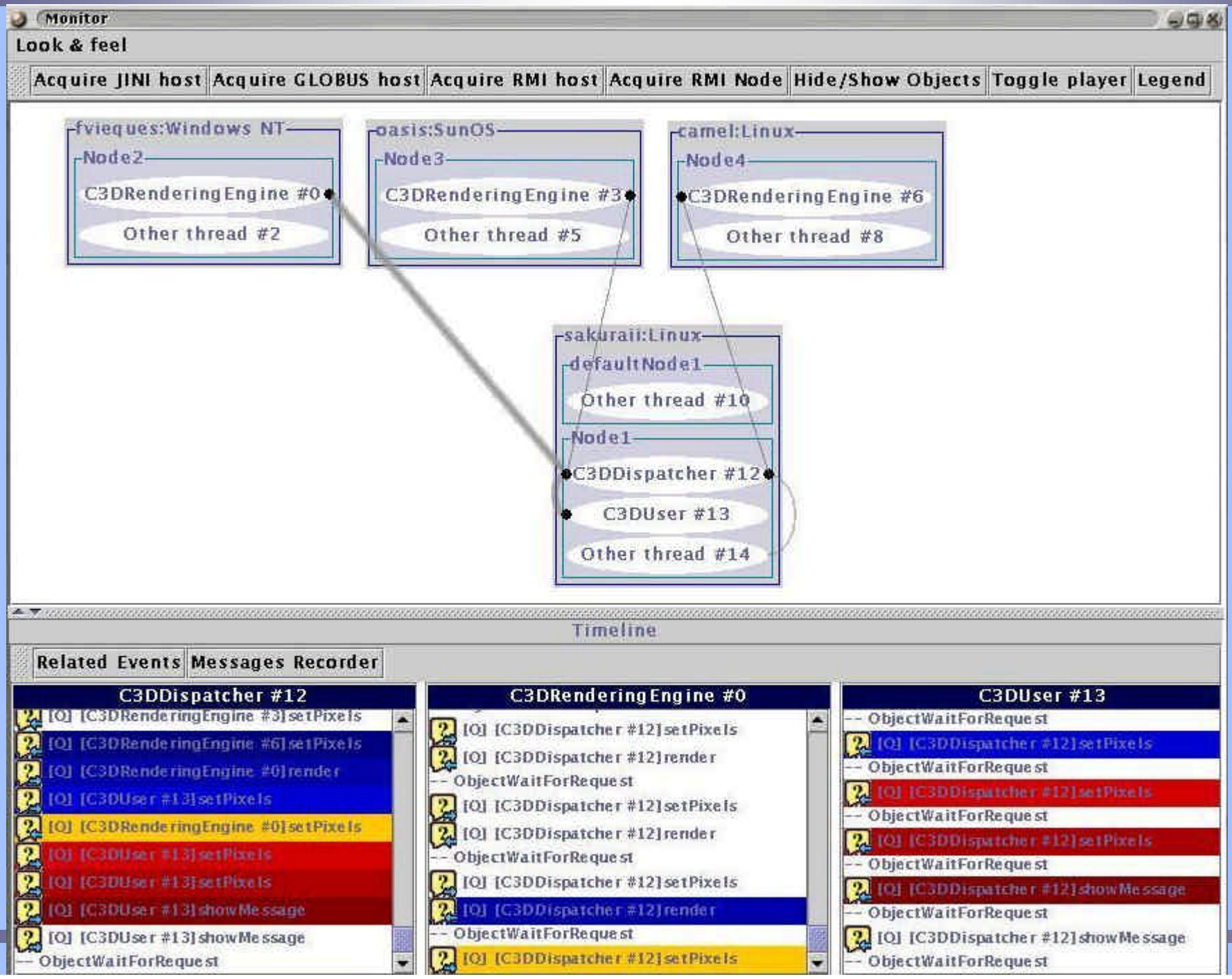
ProActive : Deployment of an application

- **Need to launch on each host:**
  - **a RMI registry**
  - **a Java Virtual Machine that executes a ProActive NODE**
    - **Have access or be able to download ProActive classes**
- **Various means to access hosts and be authenticated:**
  - **Same administrative domain (same DNS)**
  - **Have an account on a Globus host**

==> IC2D can ease the deployment phase

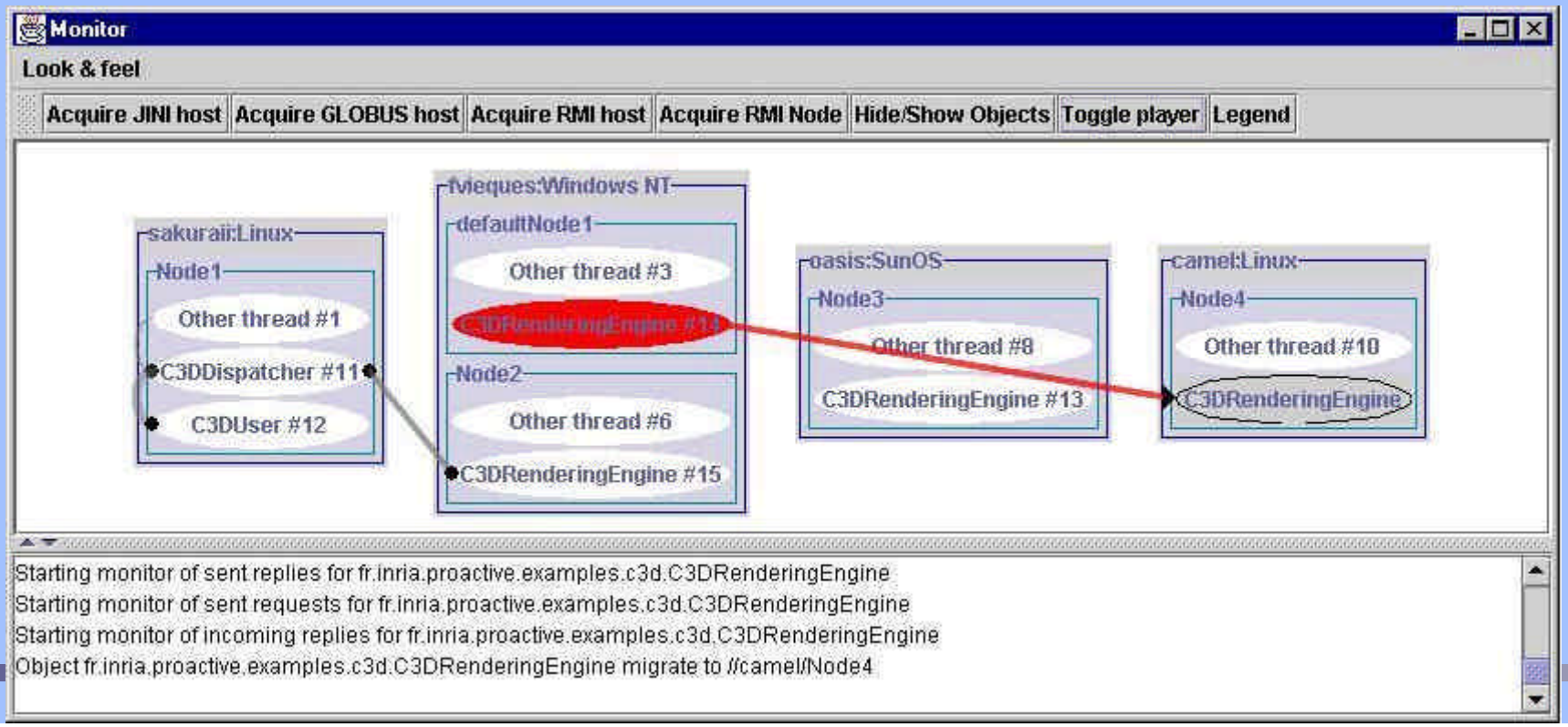
IC2D : Deployment and monitoring of an application

- **Visualisation:**
  - Hosts, JVMs, Active Objects
  - Topology: references and volume of communications
  - Status of active objects (executing, waiting, etc)
  - Migration of activities
  - Causally ordered list of remote method calls (send/receive requests, replies)
- **Deployment:**
  - Create new nodes
  - Drag-and-Drop Migration of active objects


Monitor
Look & feel
Acquire JINI host
Acquire GLOBUS host
Acquire RMI host
Acquire RMI Node
Hide/Show Objects
Toggle player
Legend
fvieques:Windows NT
Node2
C3DRenderingEngine #0
Other thread #2
oasis:SunOS
Node3
C3DRenderingEngine #3
Other thread #5
camel:Linux
Node4
C3DRenderingEngine #6
Other thread #8
sakurai:Linux
defaultNode1
Other thread #10
Node1
C3DDispatcher #12
C3DUser #13
Other thread #14
Timeline
Related Events
Messages Recorder
C3DDispatcher #12
[Q] [C3DRenderingEngine #3]setPixels
[Q] [C3DRenderingEngine #6]setPixels
[Q] [C3DRenderingEngine #0]render
[Q] [C3DUser #13]setPixels
[Q] [C3DRenderingEngine #0]setPixels
[Q] [C3DUser #13]setPixels
[Q] [C3DUser #13]setPixels
[Q] [C3DUser #13]showMessage
[Q] [C3DUser #13]showMessage
ObjectWaitForRequest
C3DRenderingEngine #0
[Q] [C3DDispatcher #12]setPixels
[Q] [C3DDispatcher #12]render
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
[Q] [C3DDispatcher #12]render
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
[Q] [C3DDispatcher #12]render
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
C3DUser #13
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
ObjectWaitForRequest
[Q] [C3DDispatcher #12]setPixels
ObjectWaitForRequest
[Q] [C3DDispatcher #12]showMessage
ObjectWaitForRequest
[Q] [C3DDispatcher #12]showMessage
ObjectWaitForRequest

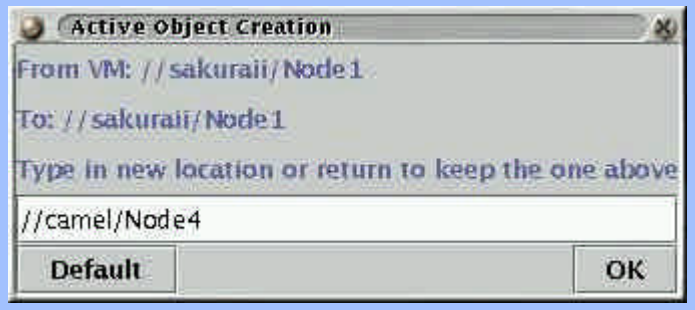
- **Deployment:**
  - Create new nodes

- Drag-and-Drop Migration of active objects



IC2D : Steering of some of the distributed aspects of applications

- Interactive control of mapping upon creation of an active object
- Interactive control of mapping upon migration of an active object

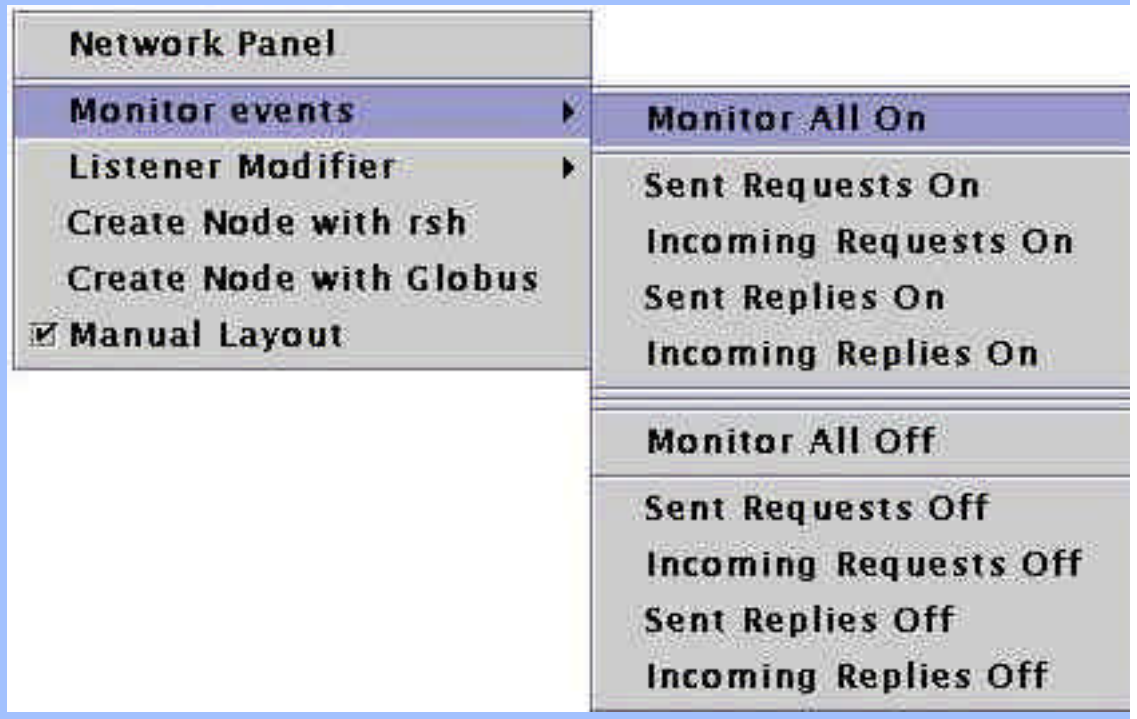


==> Modification of the execution without affecting the code

- Step by step execution of a remote method call => possible modification of its parameters (worked on)

IC2D : Design

- A unique collector: the MONITOR
- On each node, an active object acting as a SPY
- The SPY is registered as a Listener or as a Listener/Modifier for every event that could be of interest on a given or on all nodes.

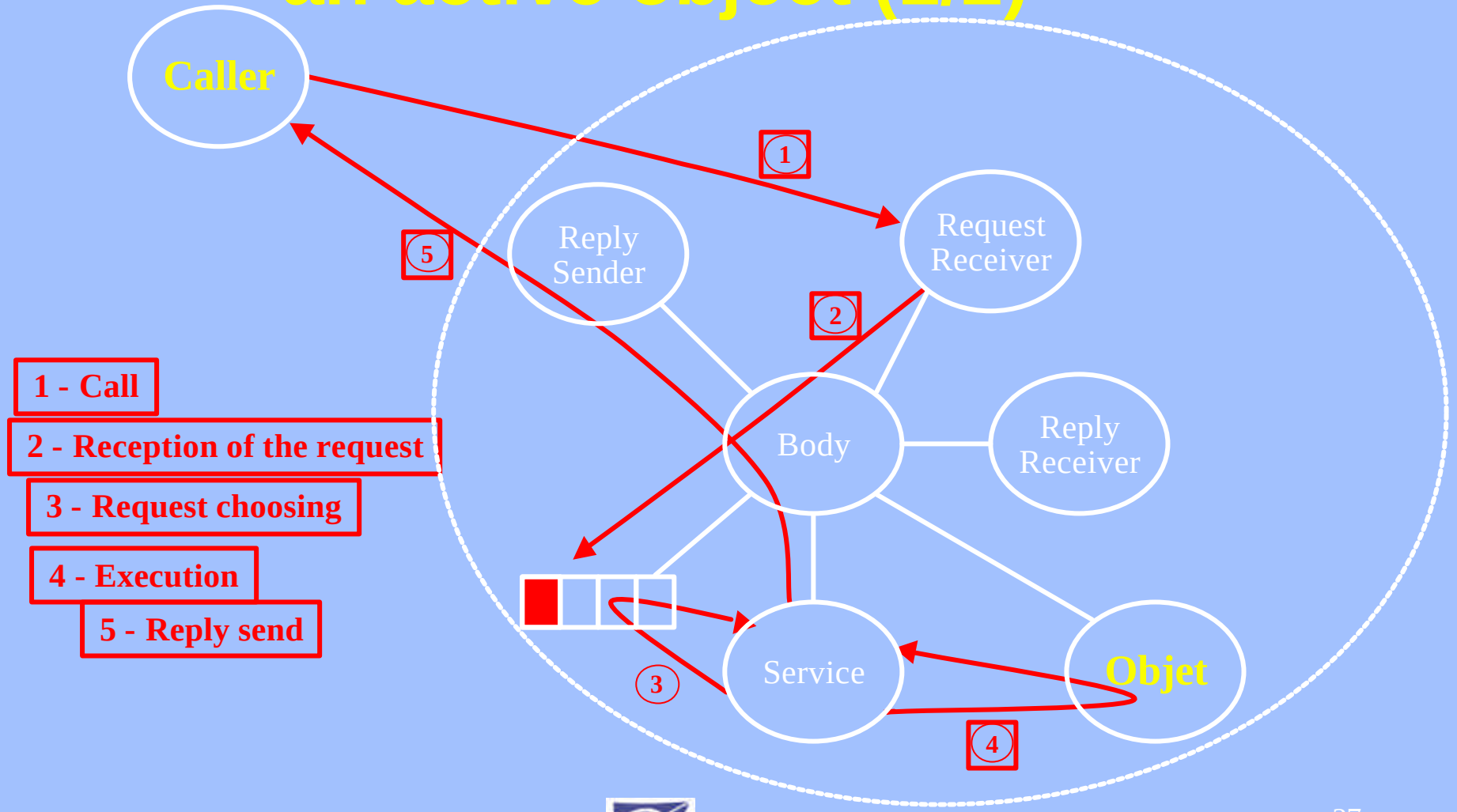


Listener

- Model “écouteur-ecouté”

- Events are generated at each important step
- then, they are possibly sent to a listener
- Those listeners may be dynamically add or removed

Exemple: Request of a service towards an active object (1/2)



1 - Call

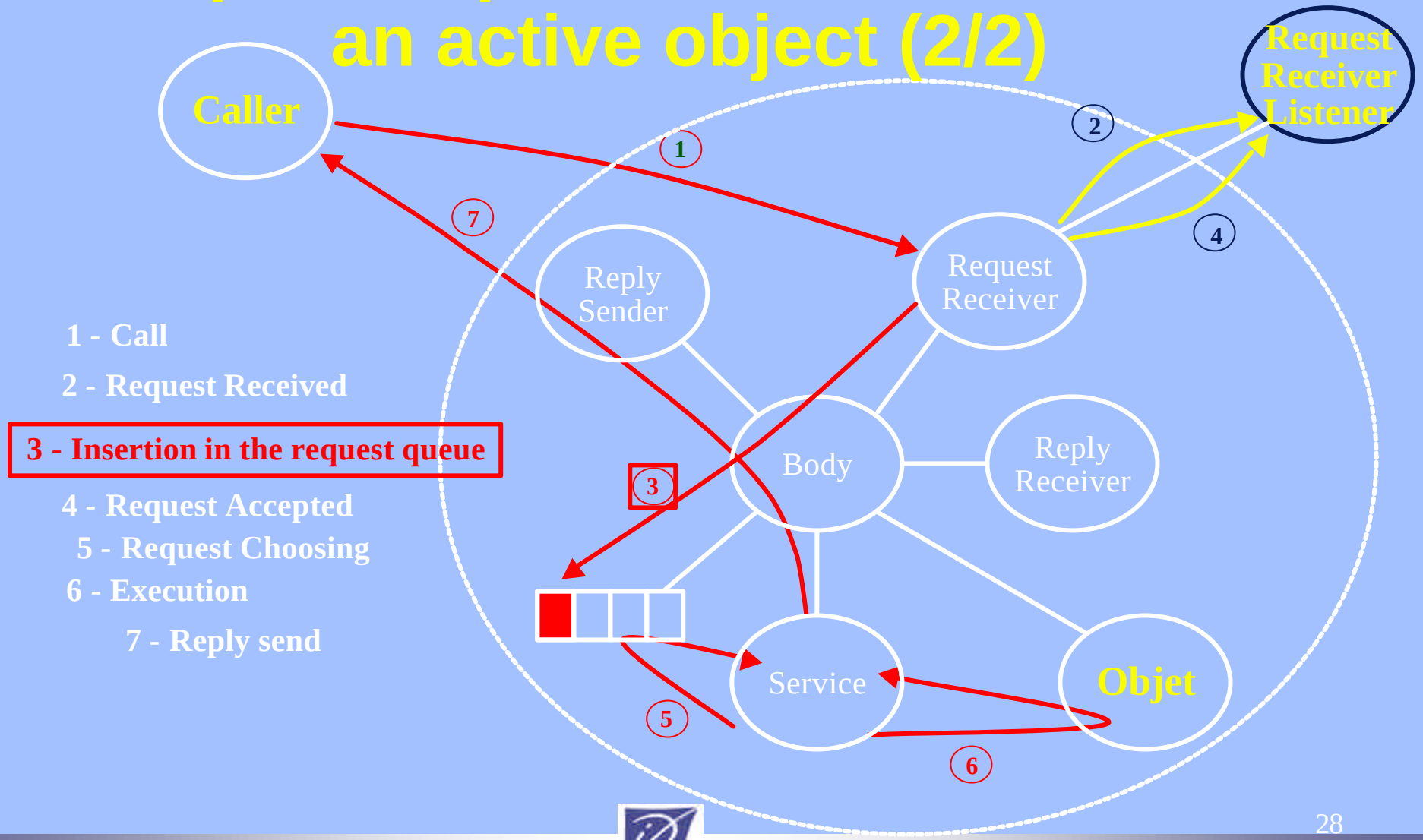
2 - Reception of the request

3 - Request choosing

4 - Execution

5 - Reply send

Exemple: Request of a service towards an active object (2/2)



1 - Call

2 - Request Received

3 - Insertion in the request queue

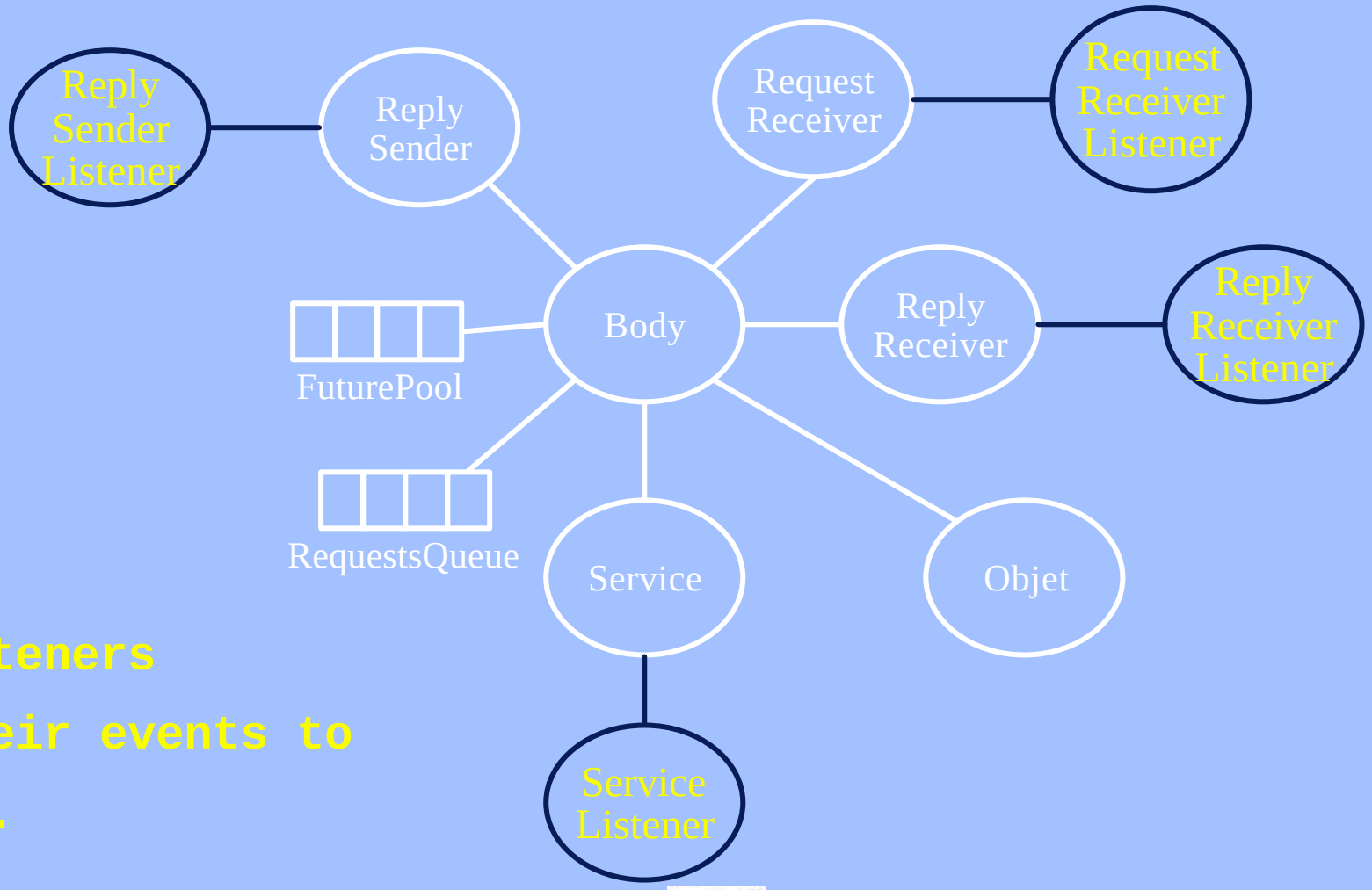
4 - Request Accepted

5 - Request Choosing

6 - Execution

7 - Reply send

Listeners at the Meta Object Level



All listeners send their events to the SPY.

3) Related Work

- **Monitoring, debugging and steering**
  - Some obvious links with xpvm, pm2, etc
  - Some works around ‘free steering’ of code
  - Some links with Java Management Extension

  ==> IC2D is really non-intrusive for the application as everything is done at the meta object level
  - Other non-functional aspects that could be monitored ?

- **Grid Computing Environments**
  - Use of the Java COG Kit in IC2D
  - Moba Threads (Java Threads that can move between Globus hosts)

  ==> IC2D provides more graphical and interactive features

  - IC2D is not a computing portal but could be included as a component
  - IC2D is not a Problem Solving Environment, is not domain specific

  ==> IC2D is currently used for EJB applications and as a basis of system and network management platform