ProActive

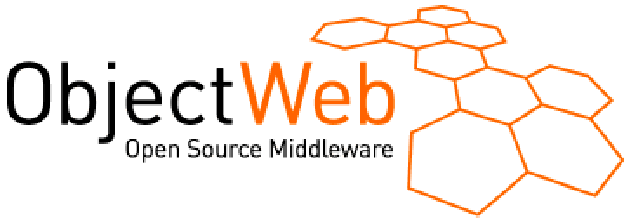
Architecture of an Open Middleware for the Grid

Romain Quilici

www.objectweb.org/ProActive

ObjectWeb Architecture meeting

July 2nd 2003



ProActive

A Java API + Tools for Parallel, Distributed Computing

- A uniform framework: **An Active Object pattern**
- A formal model behind: **Prop. Determinism, insensitivity**

to deploy.

Main features:

- Remotely accessible Objects (RMI, JINI, --> UDDI)
- Asynchronous Communications with synchro: automatic Futures
- Group Communications, Migration (mobile computations)
- XML Deployment Descriptors
- Interfaced with various protocols: rsh,ssh,LSF,Globus,--> SOAP
- Visualization and monitoring: IC2D

In the ObjectWeb Consortium

since April 2002 (LGPL License)

ObjectWeb
Open Source Middleware

Table of Contents

ProActive Runtime

- Transparently created when using ProActive
- Only one by JVM -- Singleton pattern
- Offer basics services to create or receive Active Objects
- Accessible remotely
- Partially hidden from users --> Use of Nodes (look like remote)
- ProActive Nodes are defined on PART --> possibly N nodes by JVMs
- Use of patterns to improve integration
  - Smart Proxy
  - Adapter
  - Factory ….

ObjectWeb
Open Source Middleware

ProActive Runtime Architecture
RMI case

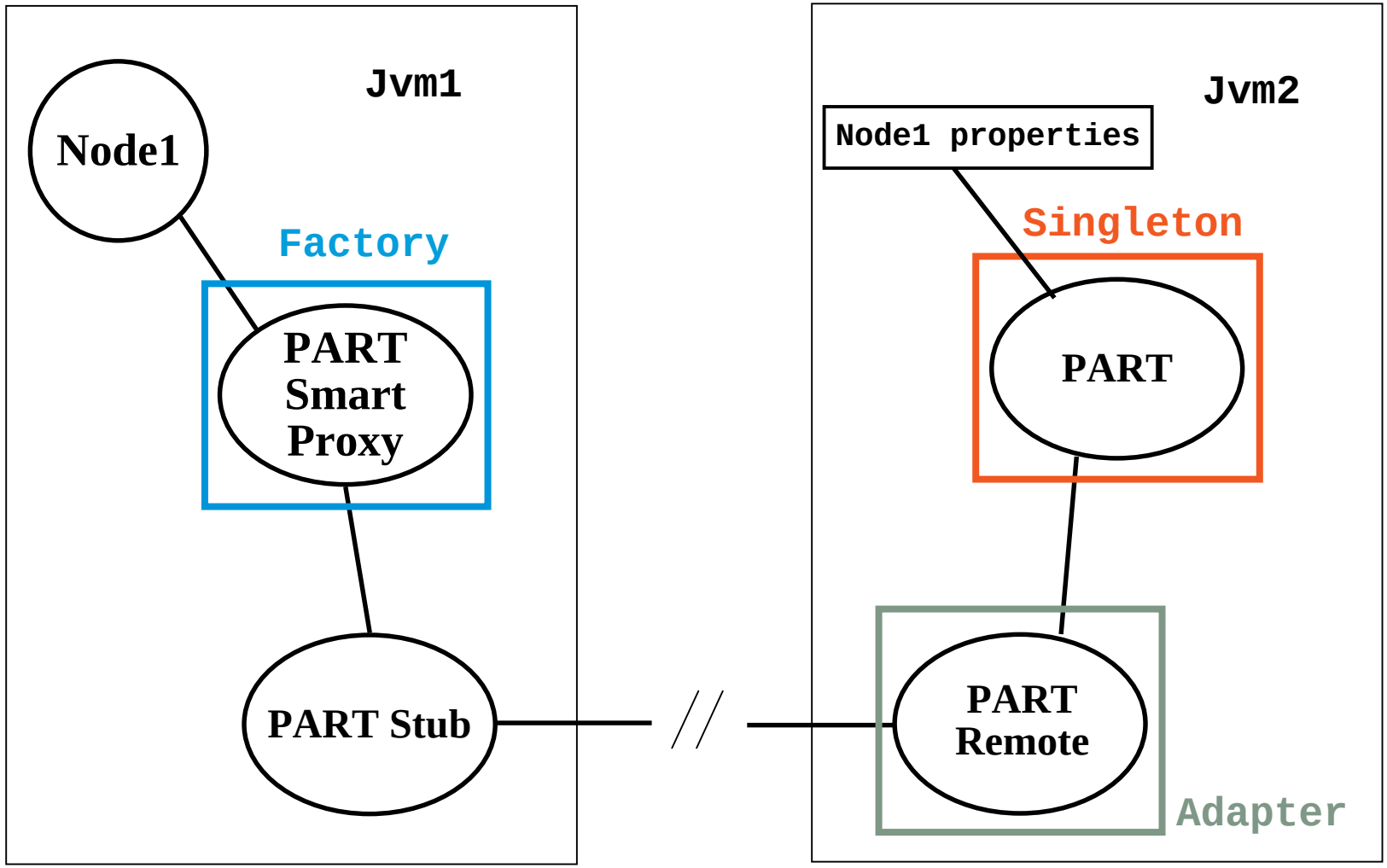
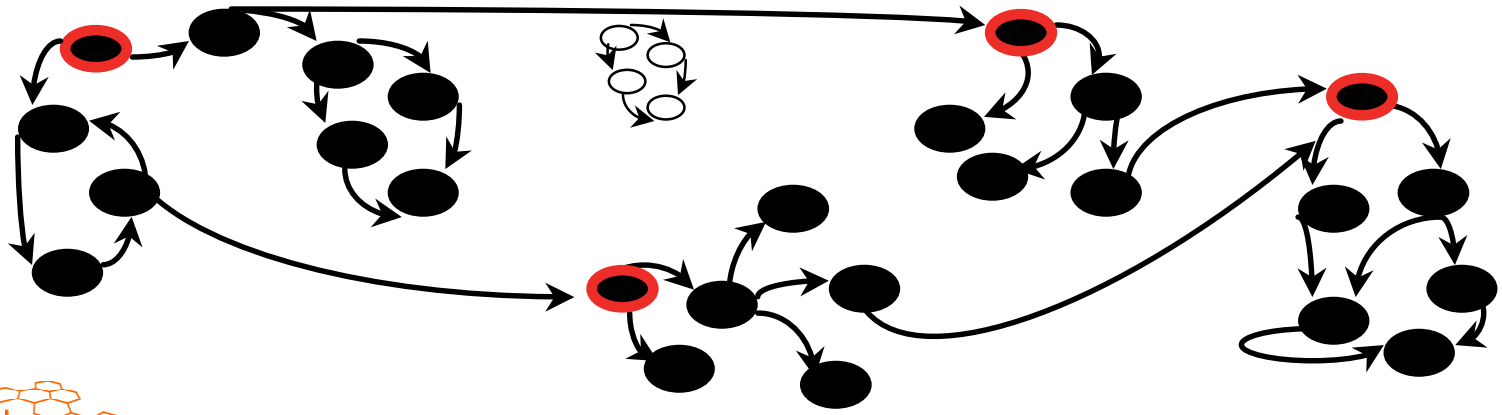


Table of Contents

ObjectWeb
Open Source Middleware

Active Object 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 (reification)



ObjectWeb
Open Source Middleware

Creating active objects

- **Instantiation-based:**

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

To get a non-FIFO behavior (Class-based):

class pA extends A implements **RunActive** { … }

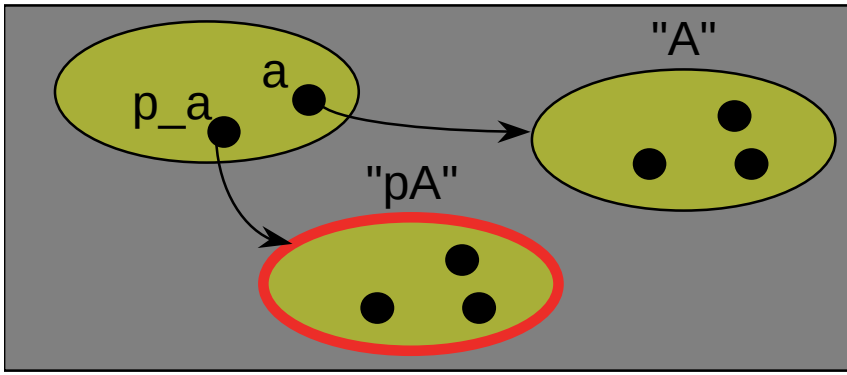
- **Object-based:**

```
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)



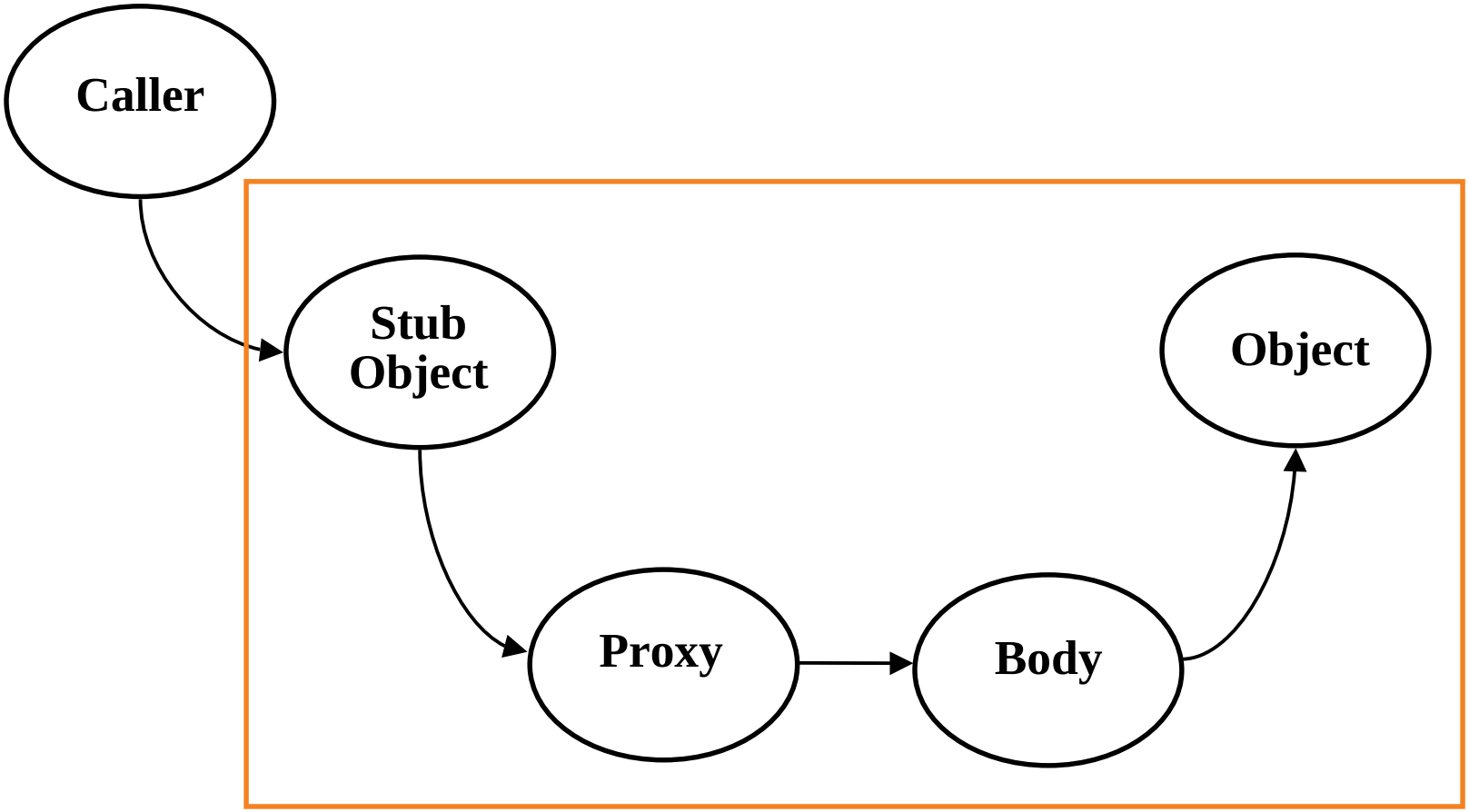

```
foo (A a)
{
  a.g (...);
  v = a.f (...);
  ...
  v.bar (...);
}
```

```
O.foo(a) : a.g()
and a.f() are
« local »
O.foo(p_a): a.g()
and a.f()are
remote + Async.»
```

- Wait-by-necessity: inter-object synchronization
  - Systematic, implicit and transparent futures (“value to come”)

    Ease the programming of synchronizations, and the reuse of routines

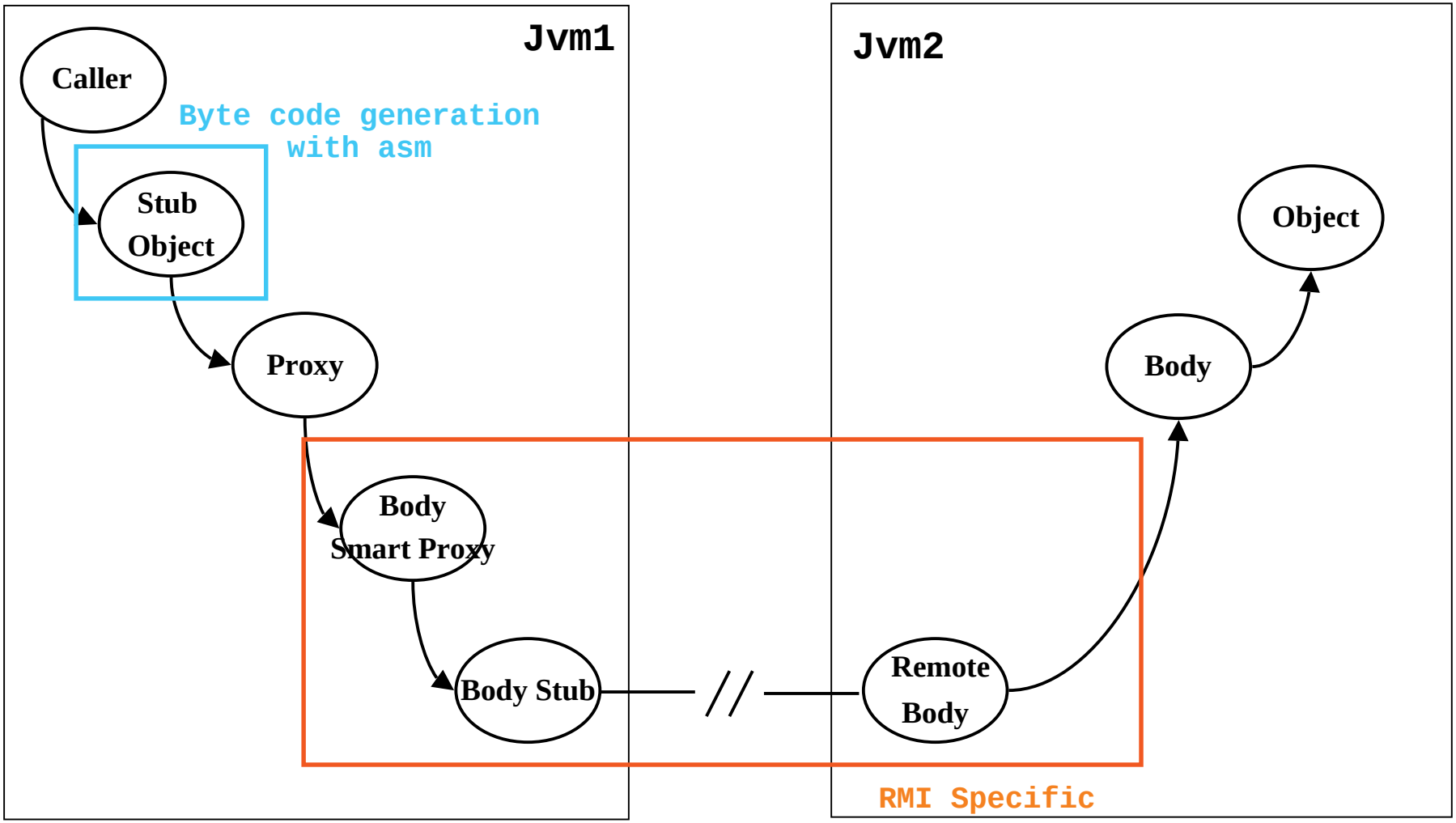
Active Object Components



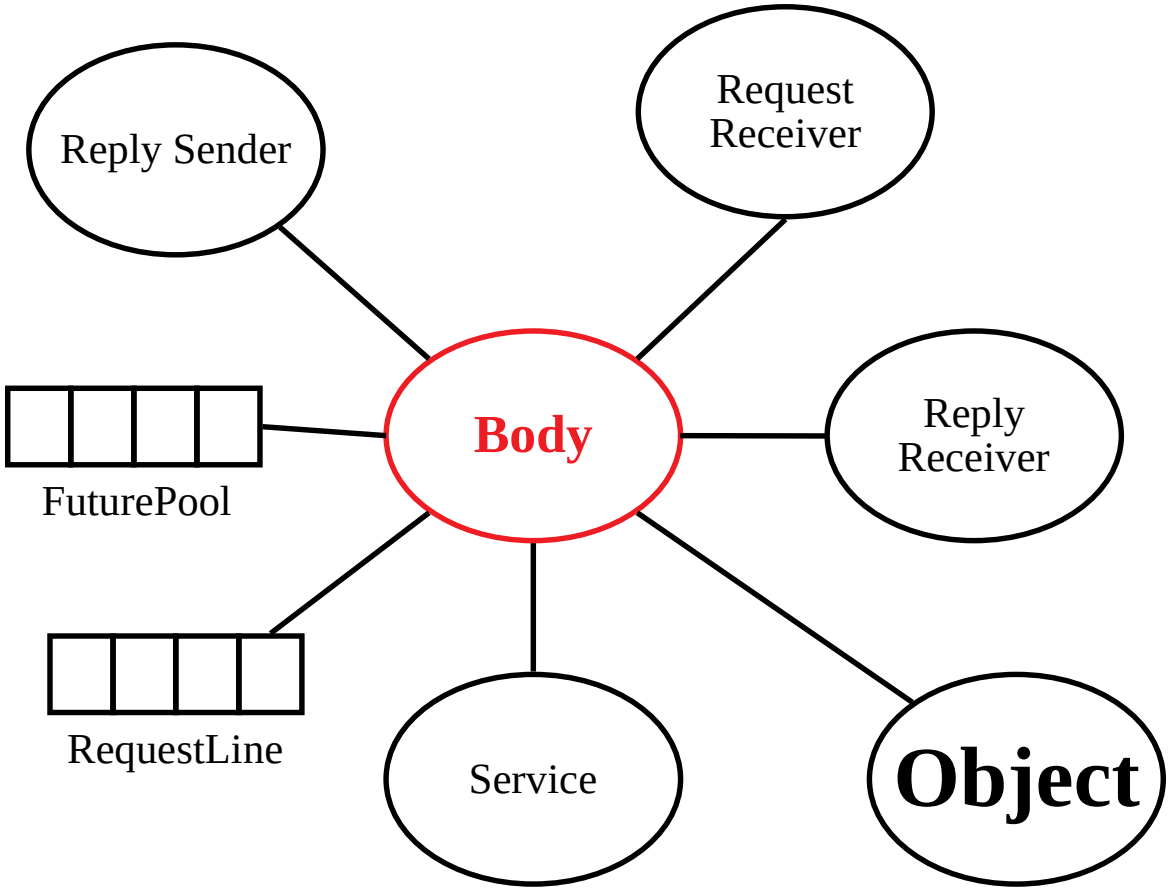
Components of an Active Object

Active Object Architecture

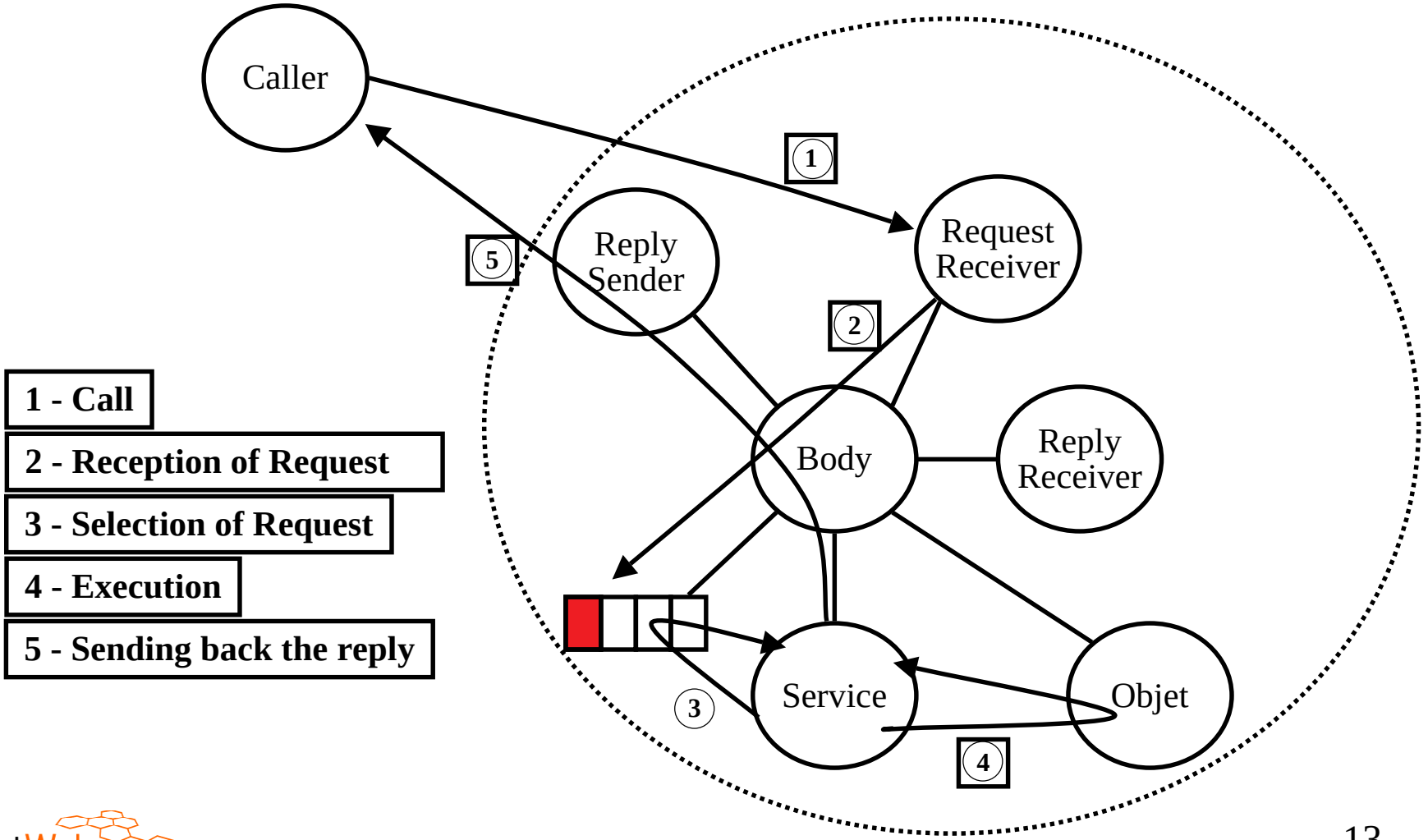
RMI case



Body Architecture



Request to an Active Object



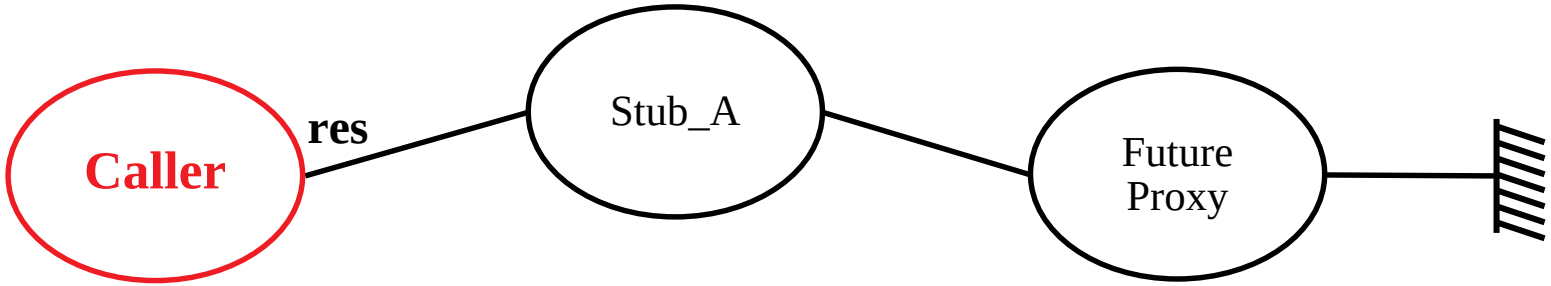
ObjectWeb
Open Source Middleware

Table of Contents

ObjectWeb
Open Source Middleware

Future Objects

```
A res = ActiveObject.foo();
```

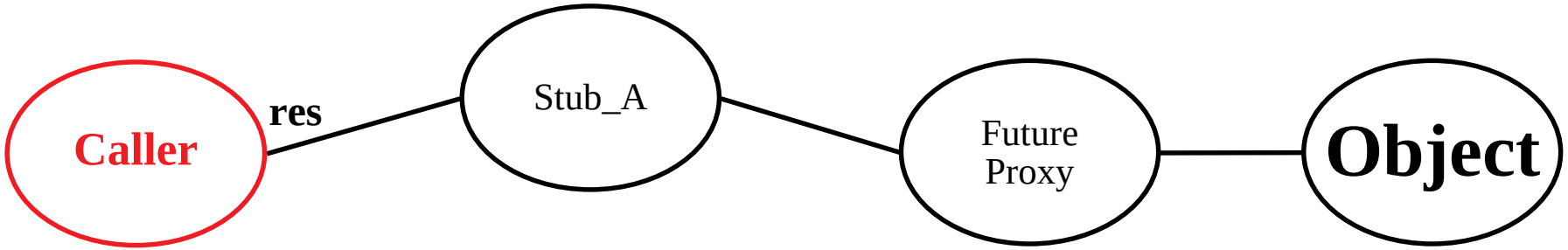


- The caller receives a Future : it continues its execution

- If it tries to access to the value or **res**, it is blocked in the future proxy (*Wait By Necessity*), until this value is available

Future Objects

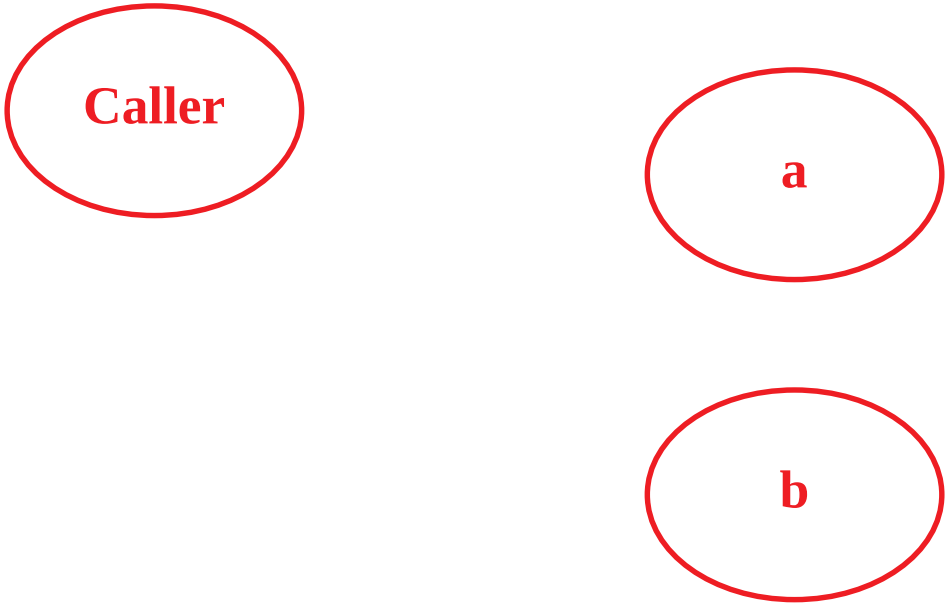
```
A res = ActiveObject.foo();
```



- When the called object finish the computation of res, the value is returned to the caller

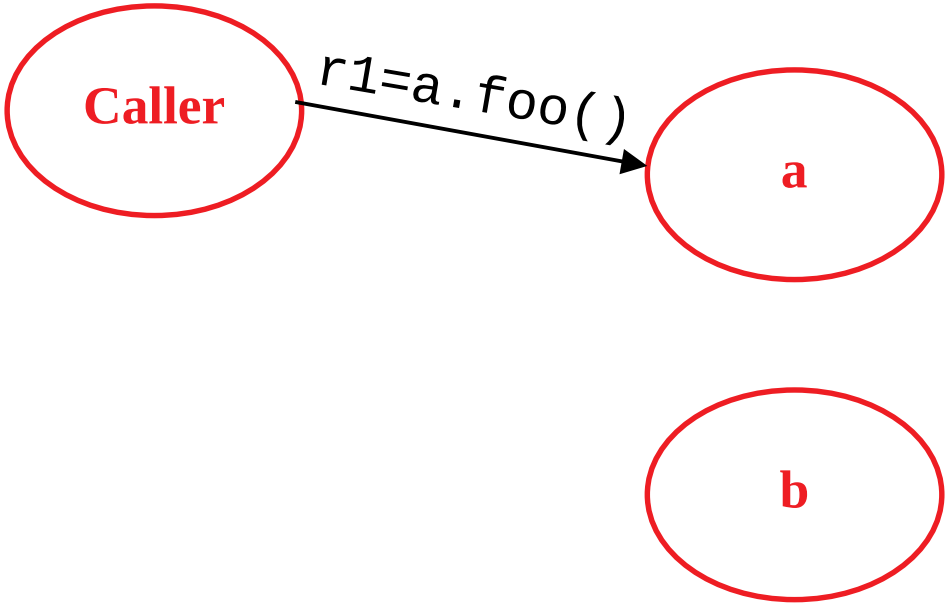
- Future is updated transparently

Automatic Continuation



Automatic Continuation

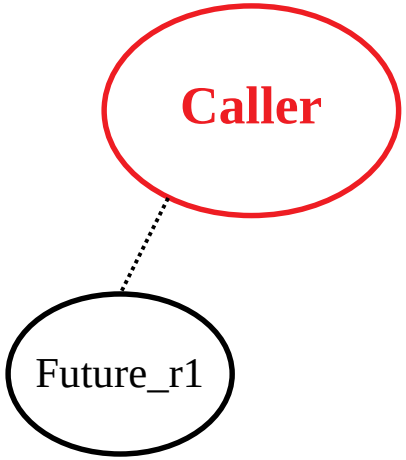
A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.

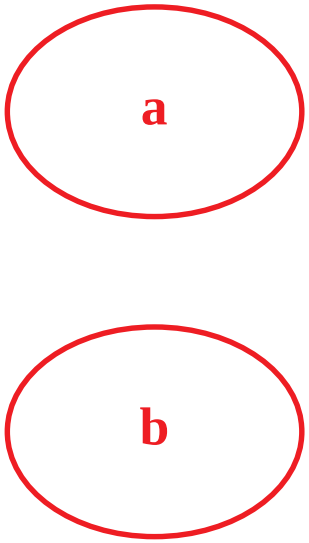



```
…
r1=a.foo()
…
```

Automatic Continuation

A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.

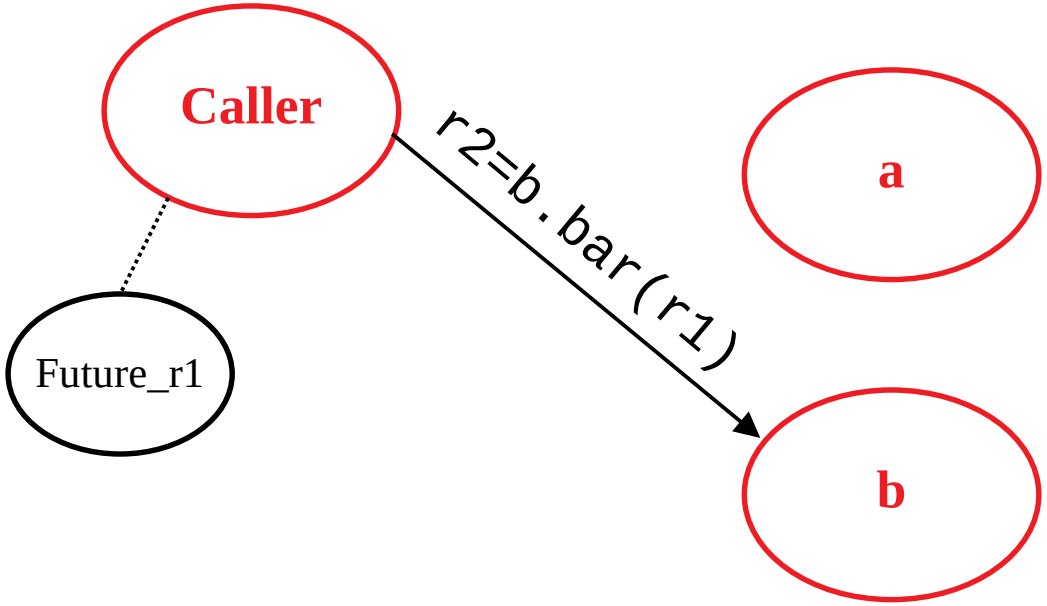





```
…
r1=a.foo()
…
```

Automatic Continuation

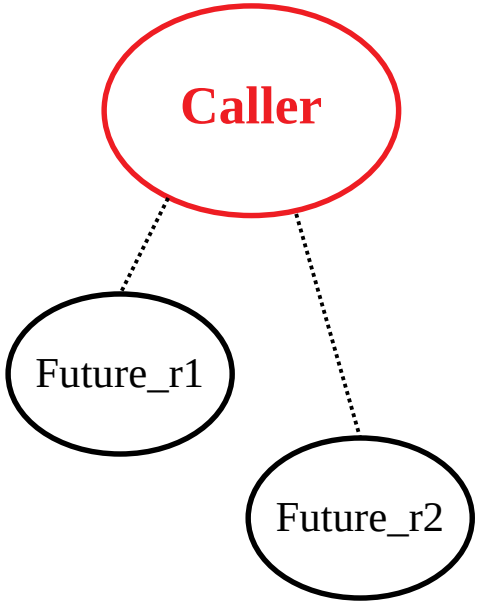
A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.

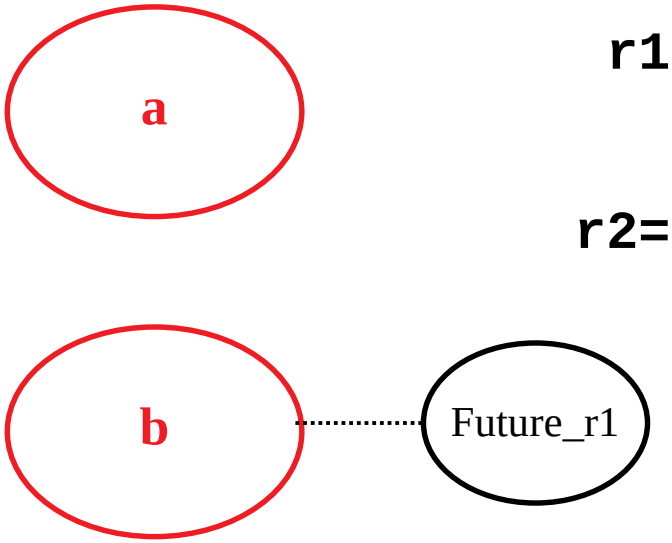



```
…
r1=a.foo()
…
r2=b.bar(r1)
```

Automatic Continuation

A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.

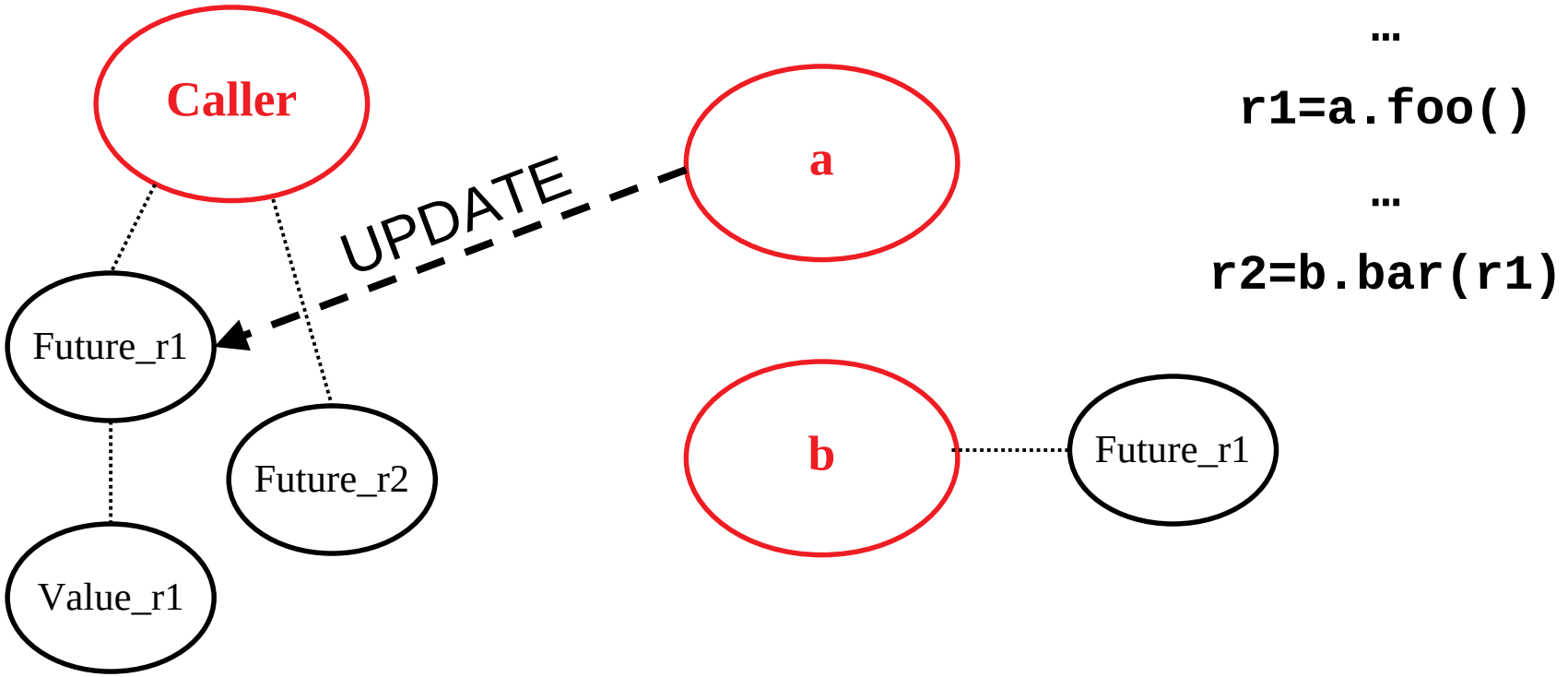





```
…
r1=a.foo()
…
r2=b.bar(r1)
```

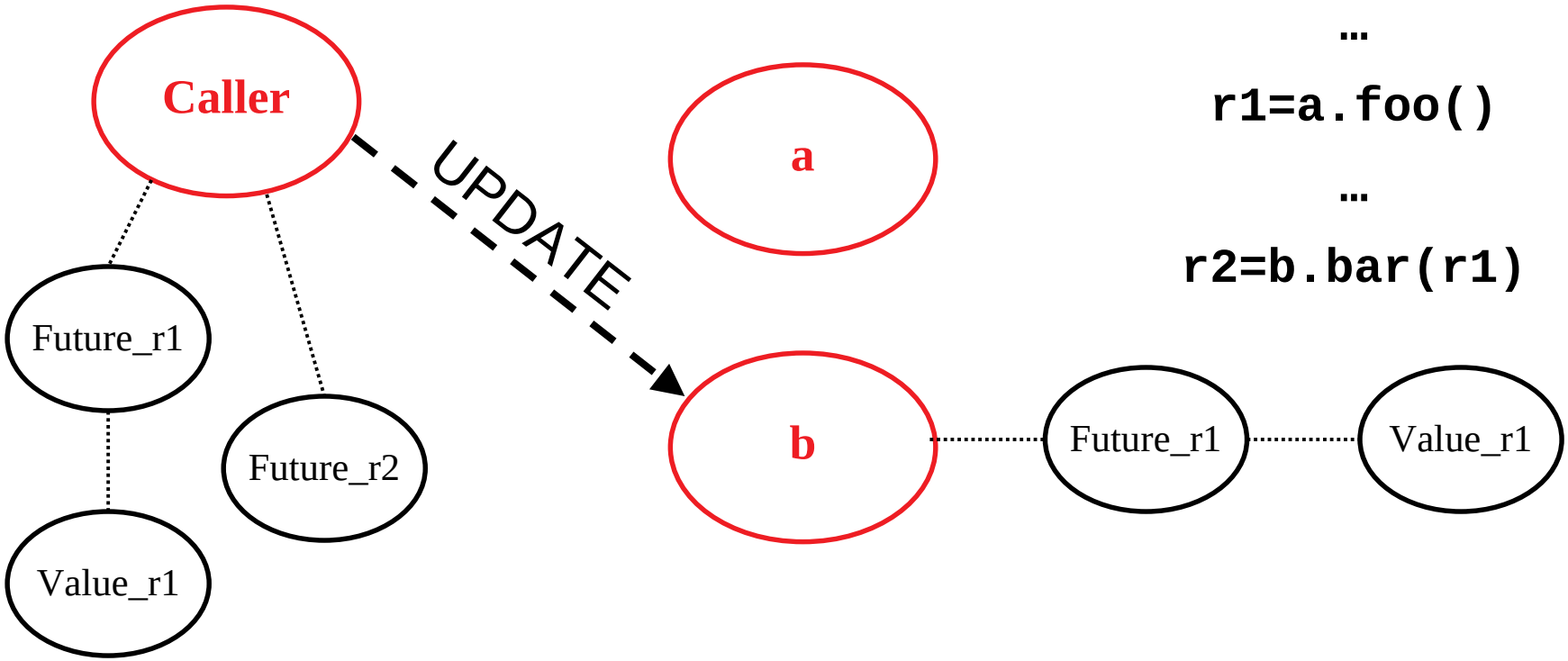
Automatic Continuation

A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.



Automatic Continuation

A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.



Automatic Continuation

A Future can be passed by parameter or by result: chains of futures will be updated by **Automatic Continuation**.

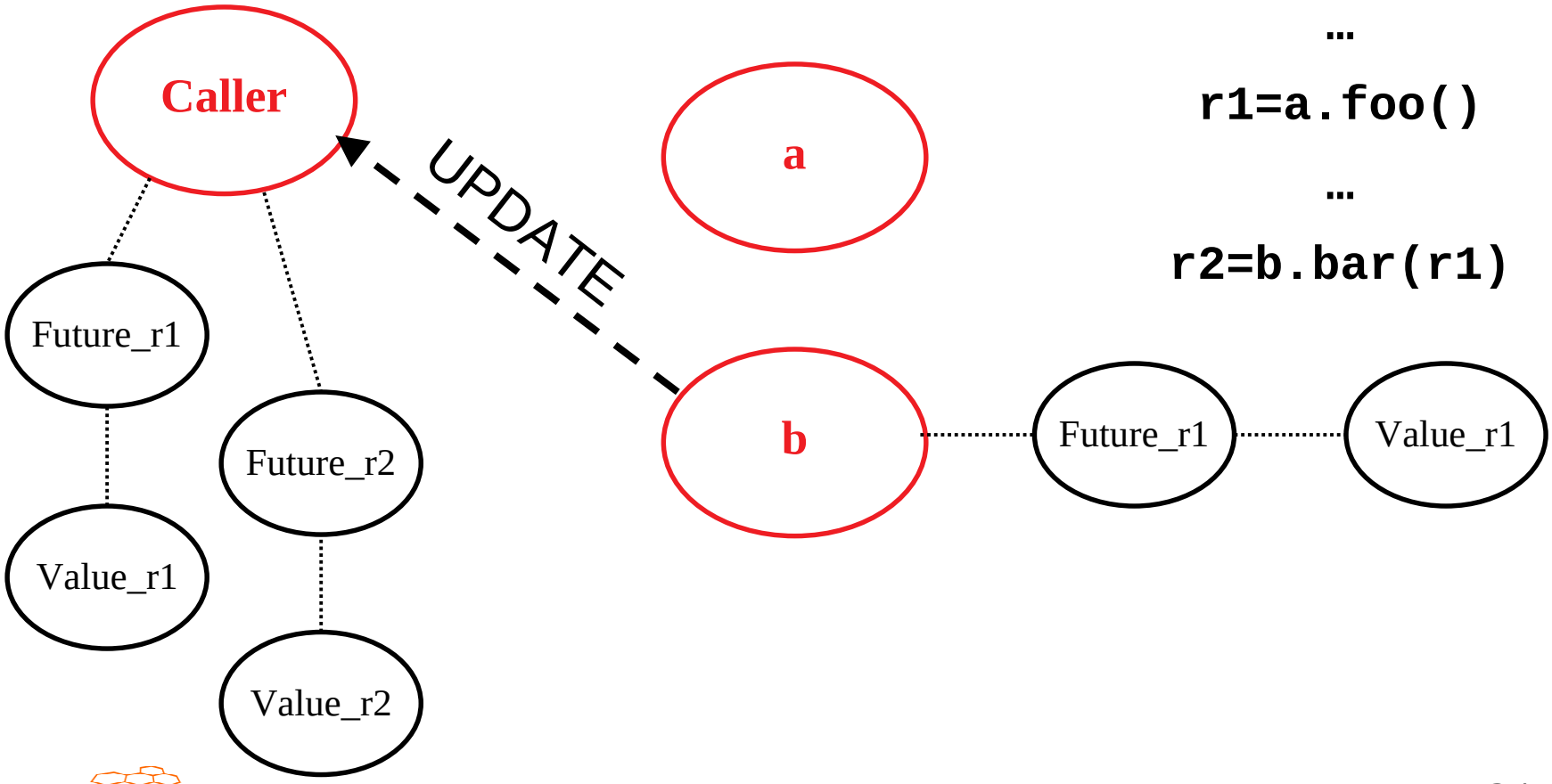


Table of Contents

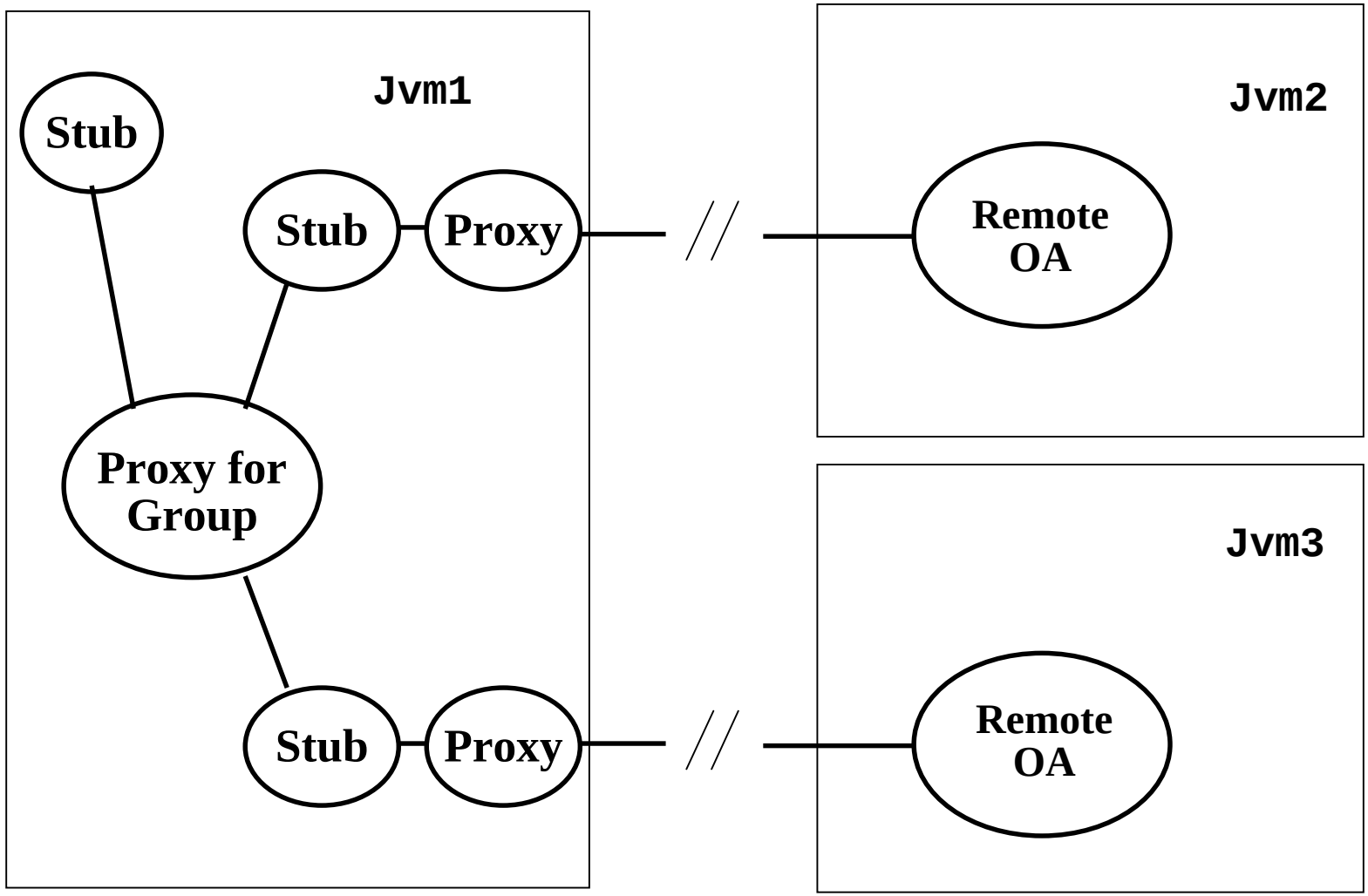
ObjectWeb
Open Source Middleware

Group Communication

- **Manipulate groups of Active Objects in a simple and typed manner**
  - **Typed groups of active and remote objects.**
  - **Maintain the ‘dot ’ notation, language property**
  - **Dynamic generation of groups of results**
- **Be able to express high-level collective communication**
  - **broadcast**
  - **scatter, gather**
- **Based on the ProActive communication mechanism**
  - **Replication of N ‘ single ’ communications**
  - **Preservation of the « rendez-vous »**
  - **Asynchronous**

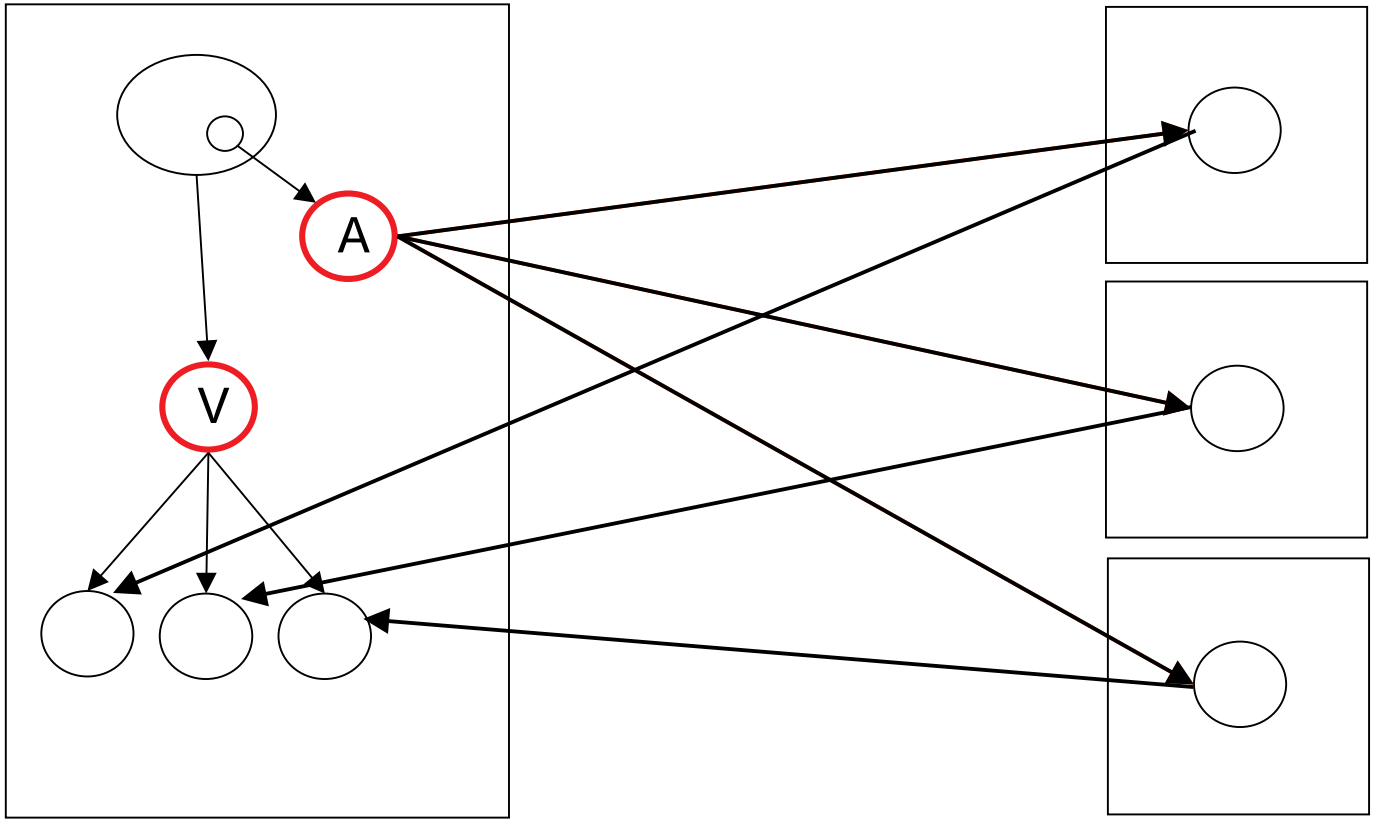
ObjectWeb
Open Source Middleware

Group Structure



Construction of a Result Group

```
A ag = newActiveGroup (…)
V v = ag.foo(param);
v.bar();
```



Typed Group

Java or Active Object

ObjectWeb
Open Source Middleware

Table of Contents

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:

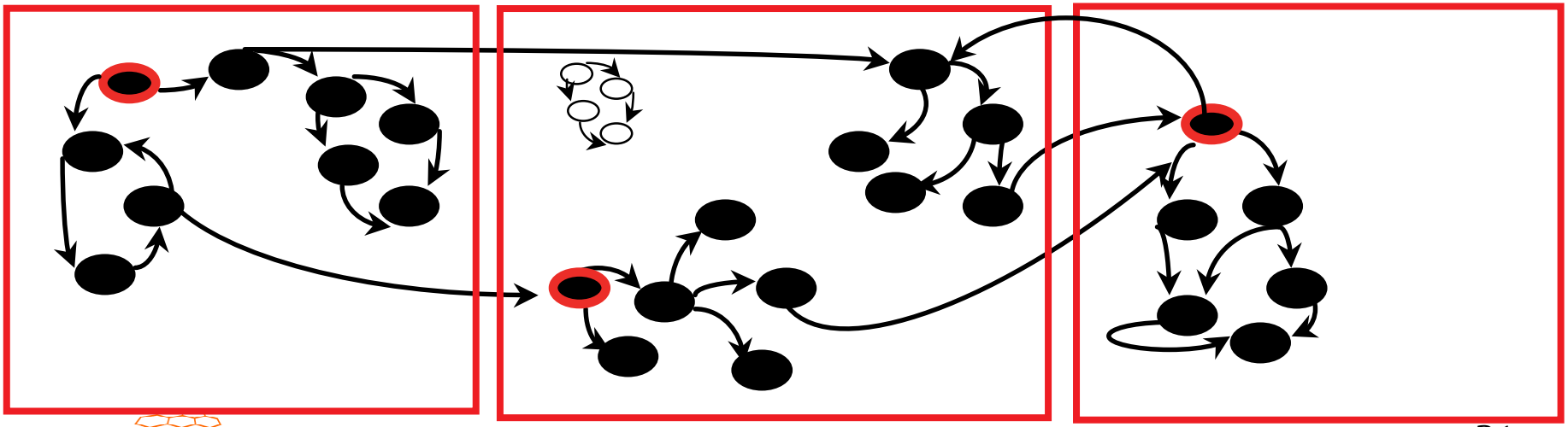
- all pending requests
- all its passive objects
- all its future objects

Automatic and transparent forwarding of:

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

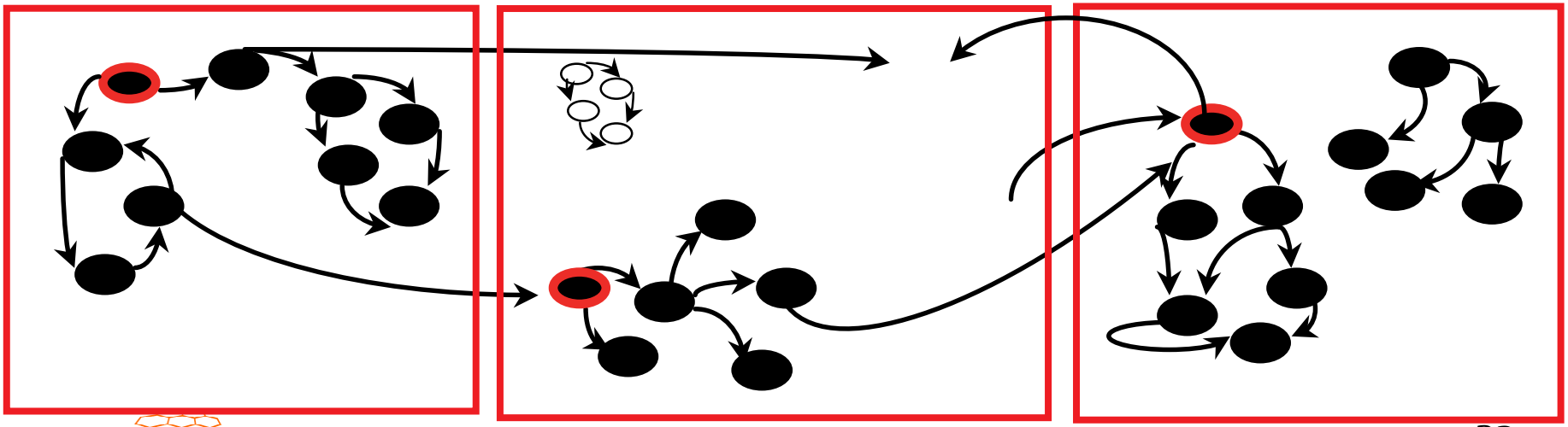
Characteristics and optimizations

- 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)



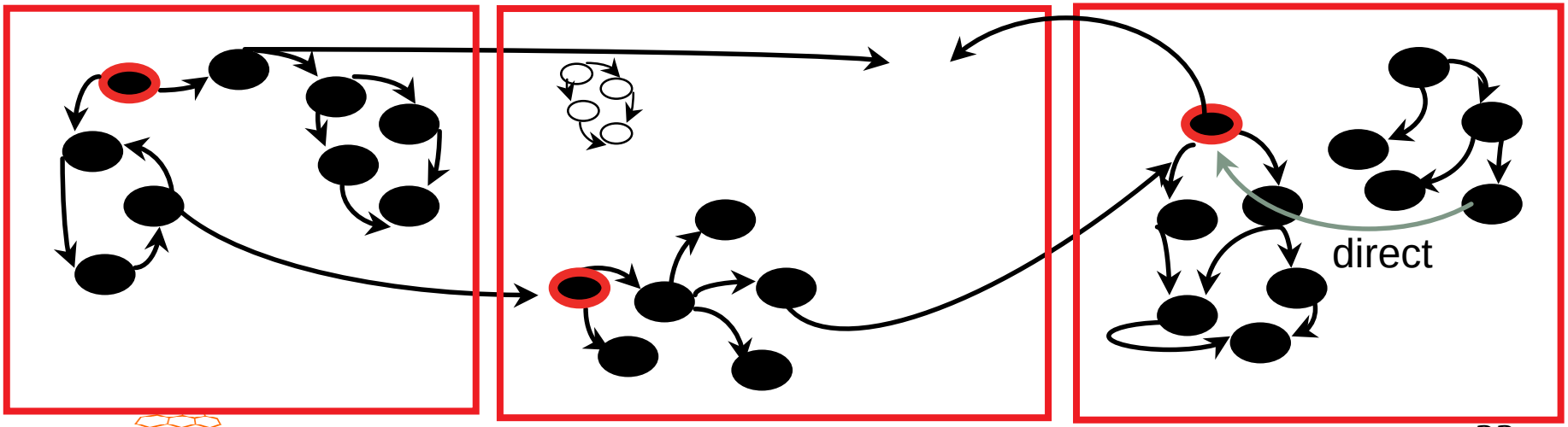
Characteristics and optimizations

- 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)



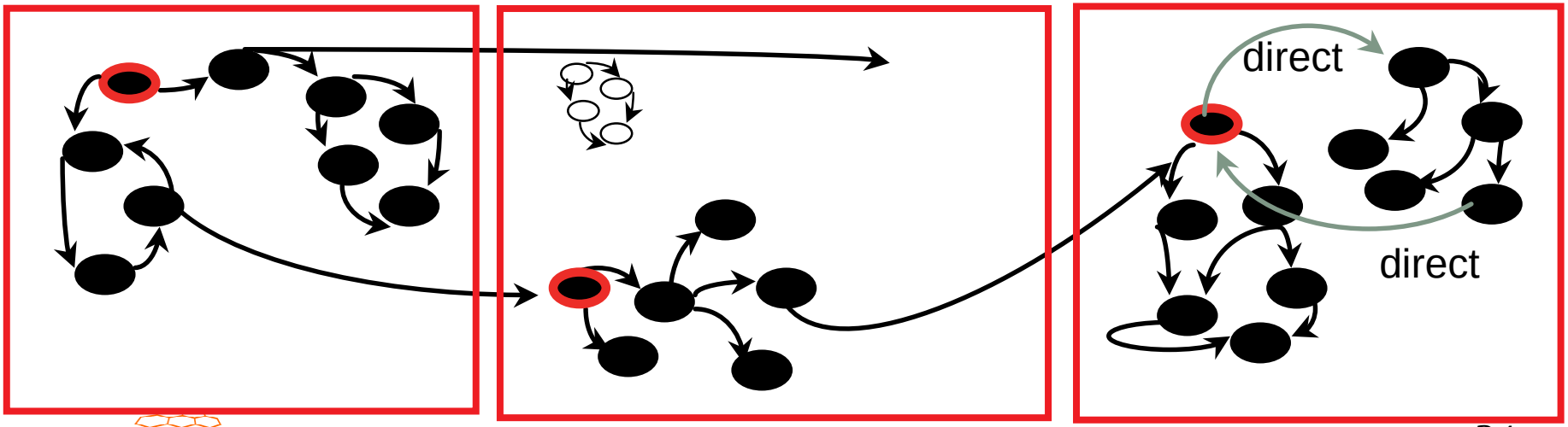
Characteristics and optimizations

- 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)



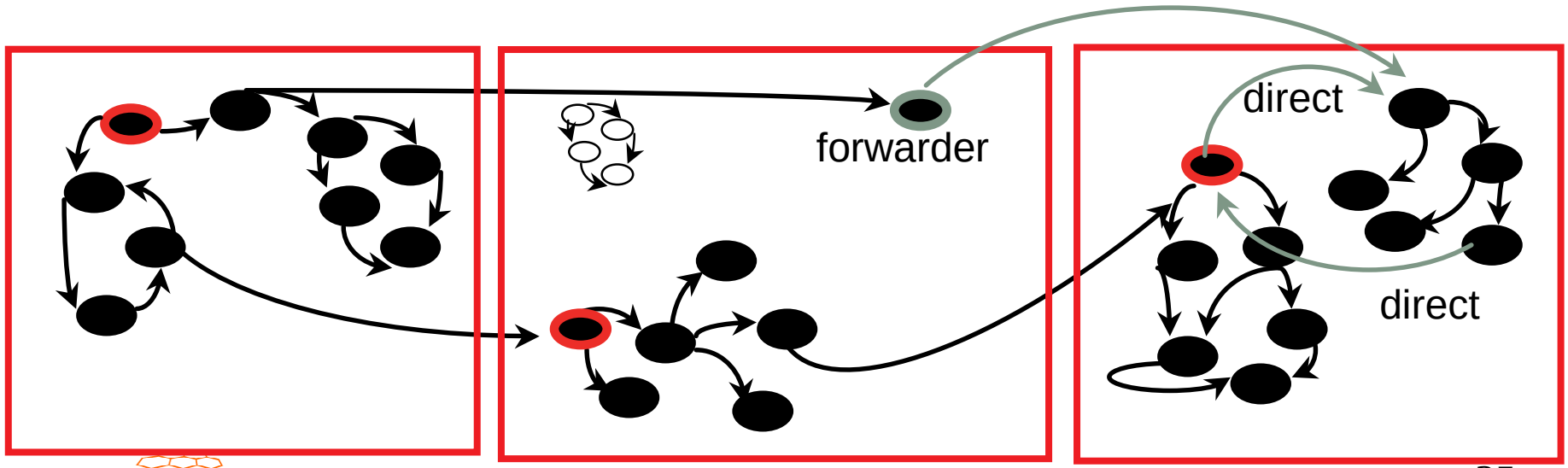
Characteristics and optimizations

- 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)



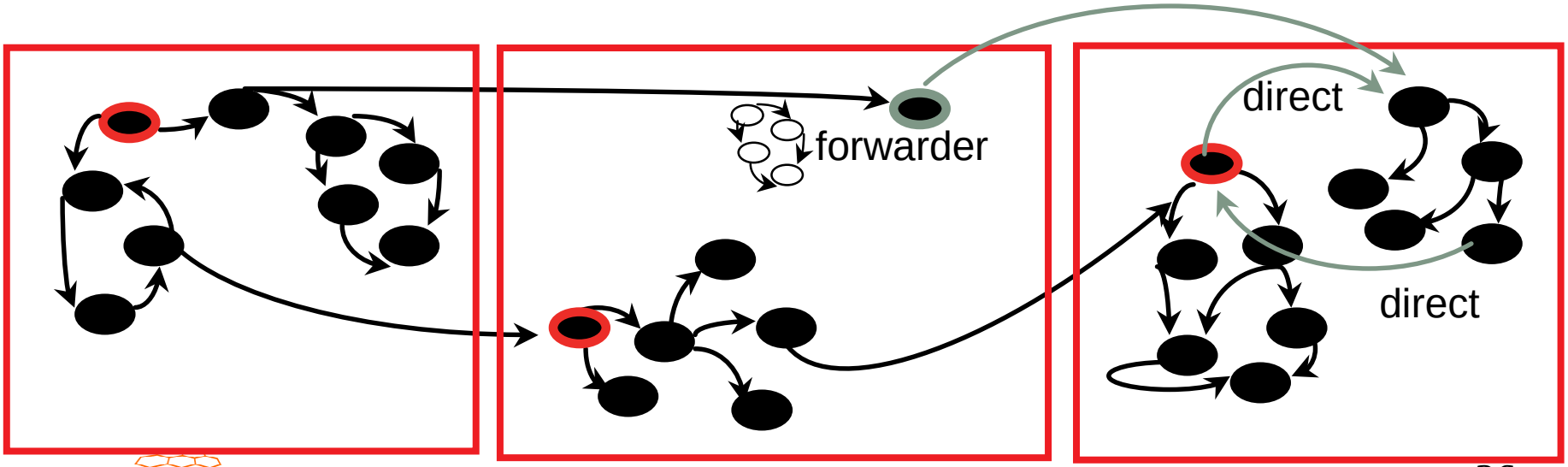
Characteristics and optimizations

- 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)



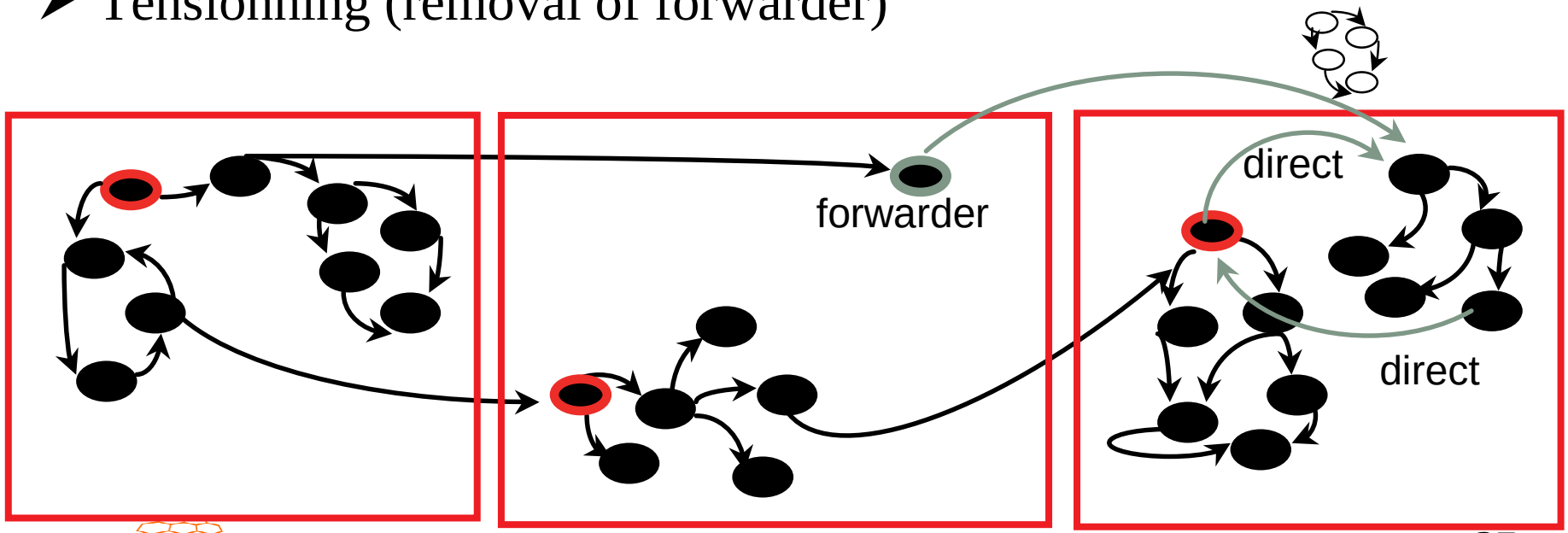
Characteristics and optimizations

- 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)



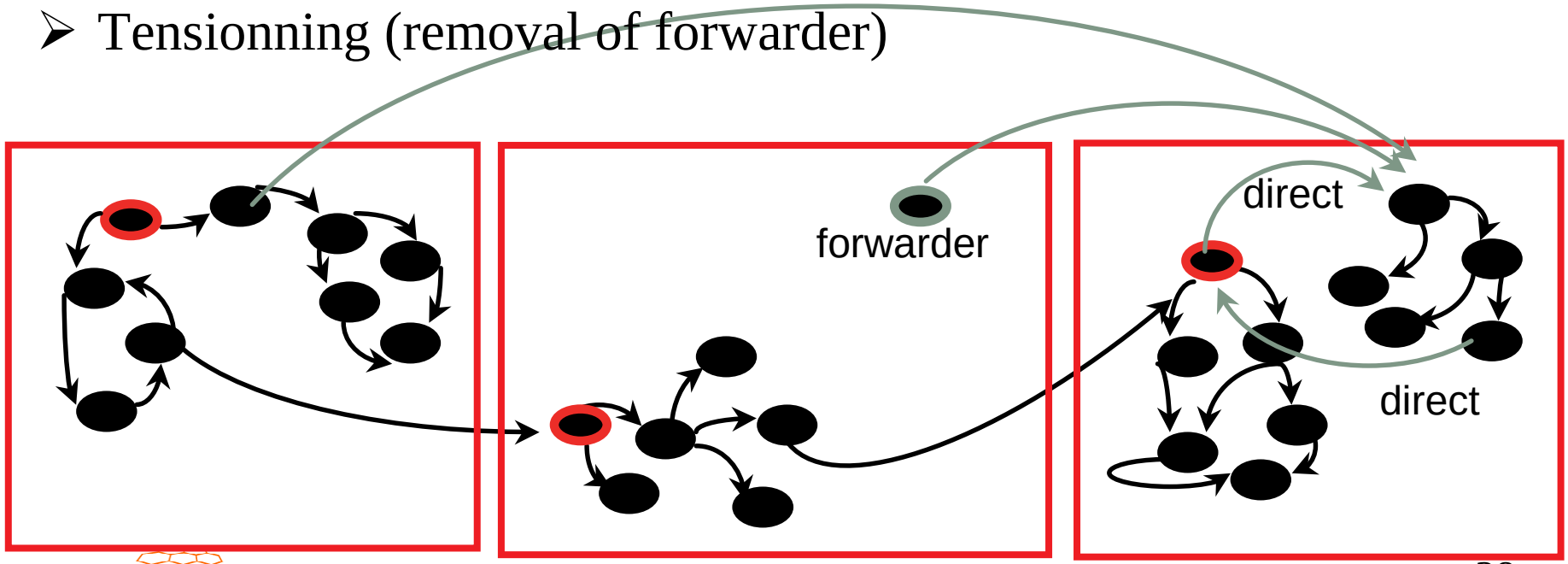
Characteristics and optimizations

- 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)



API for Mobile Agents

- Basic primitive: migrateTo
  - public static void migrateTo (String u)
  - public static void migrateTo (Node n)

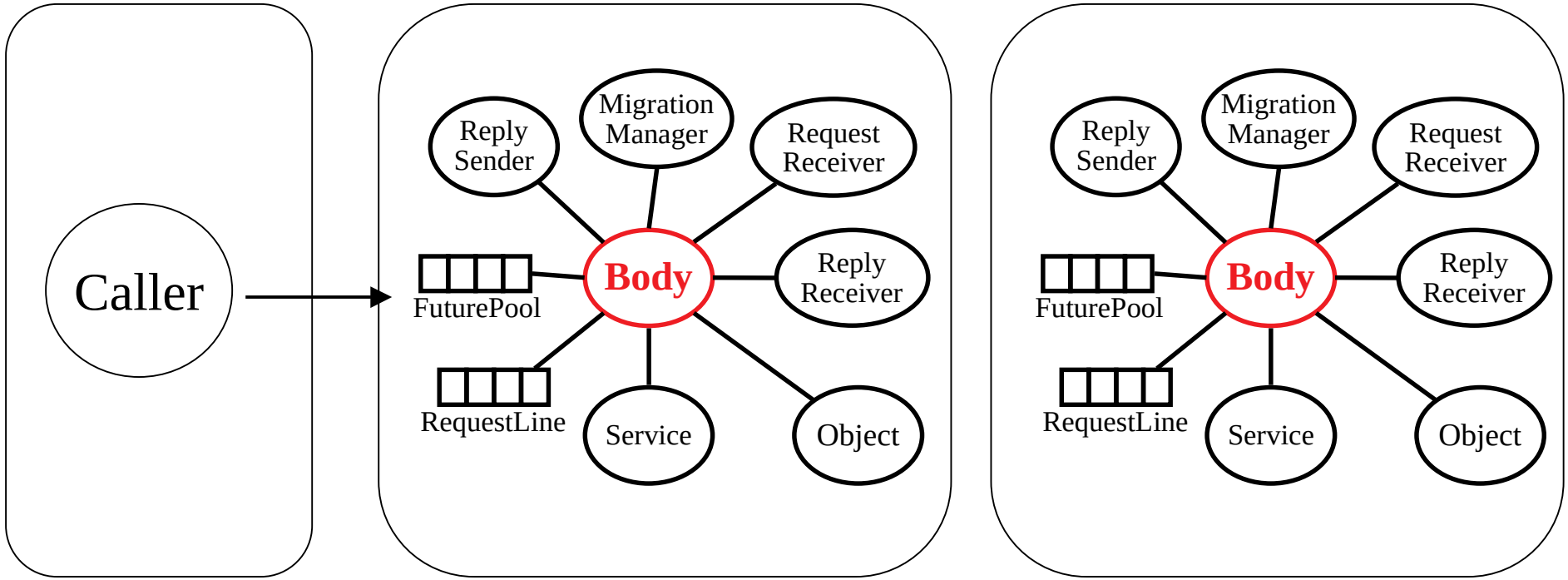
    // String or ProActive node (VM)
  - public static void migrateTo (Object o)

    // joinning another active object
- Primitive to automatically execute action upon migration
  - public static void onArrival (String r)

    // execute r upong arrival on a new Node
  - public static void onDeparture (String r)

ObjectWeb
Open Source Middleware

Migration


Caller
Reply Sender
Migration Manager
Request Receiver
FuturePool
Body
Reply Receiver
RequestLine
Service
Object
Reply Sender
Migration Manager
Request Receiver
FuturePool
Body
Reply Receiver
RequestLine
Service
Object

Migration

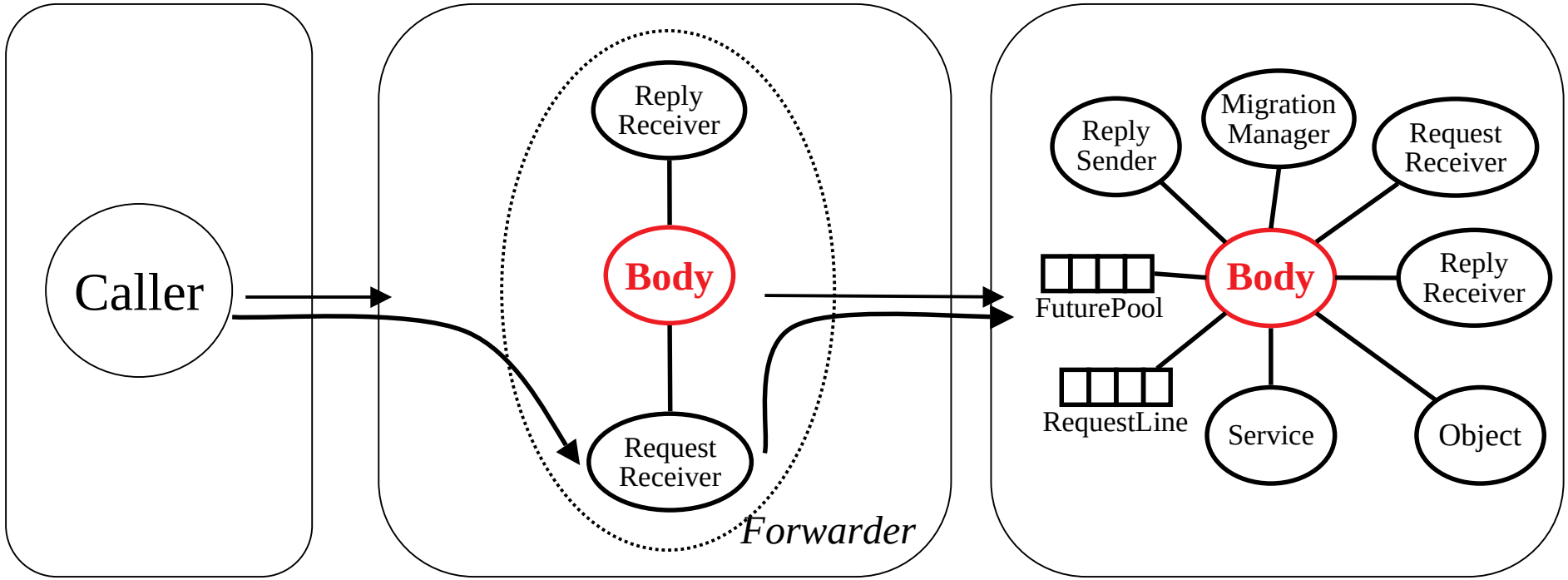

Caller
Reply Receiver
Body
Request Receiver
Forwarder
Reply Sender
Migration Manager
Request Receiver
FuturePool
Body
Reply Receiver
RequestLine
Service
Object

Table of Contents

ObjectWeb
Open Source Middleware

Abstract Deployment Model Objectives

- Problem:
  - Difficulties and lack of flexibility in deployment
  - Avoid scripting for: configuration, getting nodes, connecting, etc.
- A key principle:
  - **Abstract Away from source code:**
    - Machines
    - Creation Protocols
    - Lookup and Registry Protocols
- Context:
  - Grid
  - Distributed Objects, Java
  - Not legacy-code driven, but adaptable to it

ObjectWeb Open Source Middleware

Descriptors: based on Virtual Nodes

- Virtual Node (VN):
  - Identified as a string name
  - Used in program source
  - Configured (mapped) in an XML descriptor file --> Nodes
- Operations specified in descriptors:
  - Mapping of VN to JVMs (leads to Node in a JVM on Host)
  - Register or Lookup VNs, Create or Acquire JVMs
  - Components Definition, Security Settings

Program Source	Descriptor (RunTime)
Activities (AO) --> VN	VN --> JVMs --> Hosts

Runtime structured entities: 1 VN --> n Nodes in n JVMs

ObjectWeb
Open Source Middleware

Mapping Virtual Nodes: example

Definitions and mapping

```
<virtualNodesDefinition>
    <virtualNode name="Dispatcher"/>
</virtualNodesDefinition>

<map virtualNode="Dispatcher">
    <jvmSet>
        <vmName value="Jvm1"/>
    </jvmSet>
</map>

<jvm name="Jvm1">
    <acquisition method="rmi"/>
    <creation>
        <processReference refid="jvmProcess"/>
    </creation>
</jvm>
```

Definition of Virtual Nodes

Mapping of Virtual Nodes

ObjectWeb Open Source Middleware

Mapping Virtual Nodes: example

Infrastructure informations

```
<processDefinition id="jvmProcess">
    <jvmProcess
    class="org.objectweb.proactive.core.process.JVMNodeProcess"/>
</processDefinition>
```

JVM on the current Host

```
<processDefinition id="rshProcess">
    <rshProcess
    class="org.objectweb.proactive.core.process.rsh.RSHJVMProcess"
                hostname="sea.inria.fr">
        <processReference refid="jvmProcess"/>
    </rshProcess>
</processDefinition>
```

JVM started using RSH

ObjectWeb Open Source Middleware

Virtual Nodes in Programs

1. Load the descriptor file

```
Descriptor pad = ProActive.getDescriptor
    ("file://ProActiveDescriptor.xml");
```

2. Activate the mapping

```
VirtualNode vn = pad.activateMapping ("Dispatcher");  //
Triggers the JVMs
```

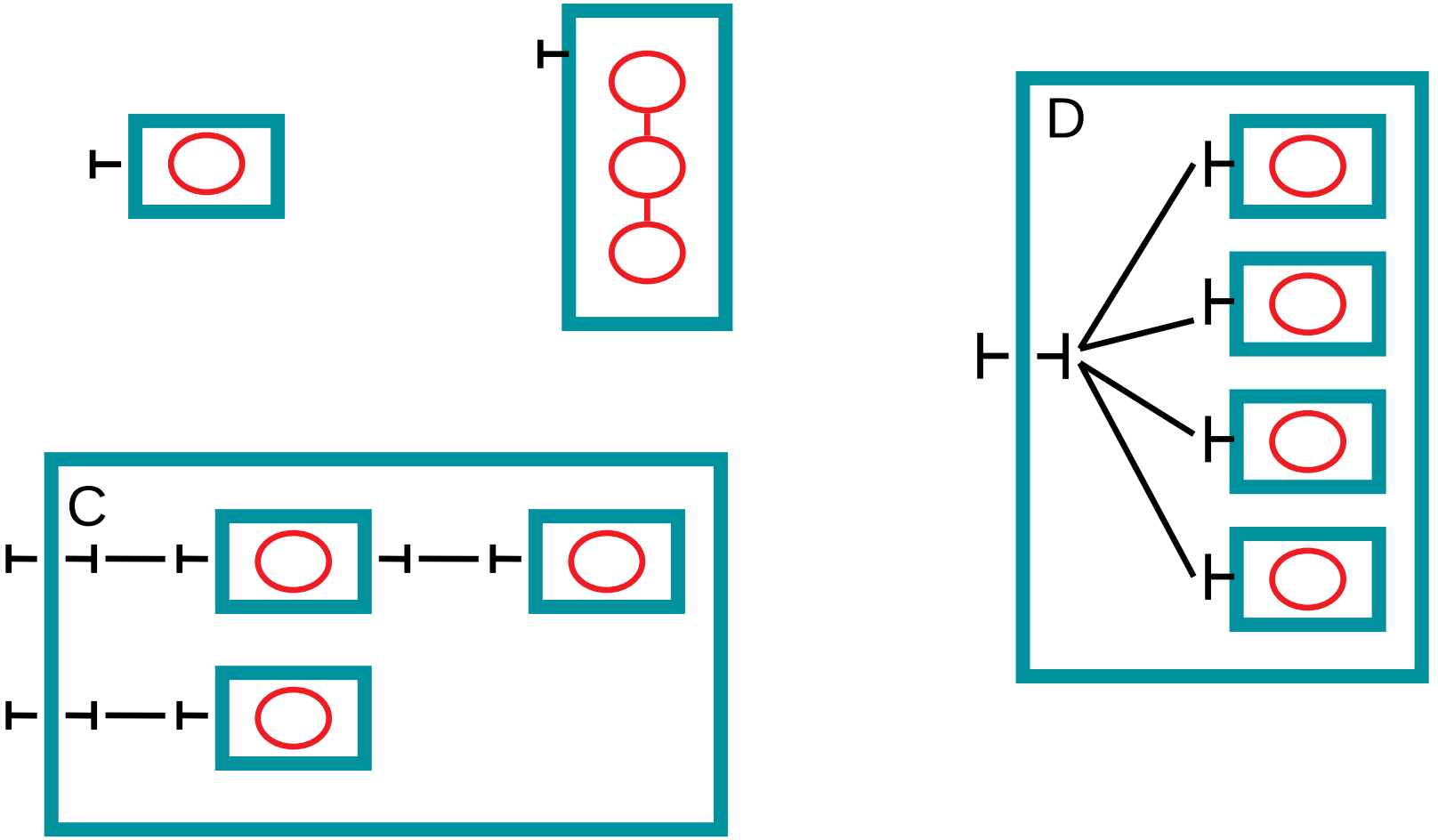
3. Use nodes

```
Node node = vn.getNode();
...
C3D c3d = ProActive.newActive("C3D", param, node);
        log (  ... "created at:  " + node.name() + node.JVM()
        + node.host() );
```

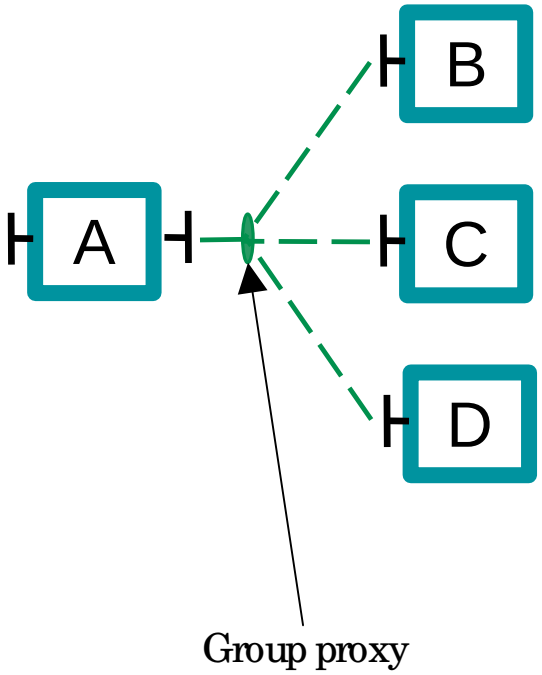
Table of Contents

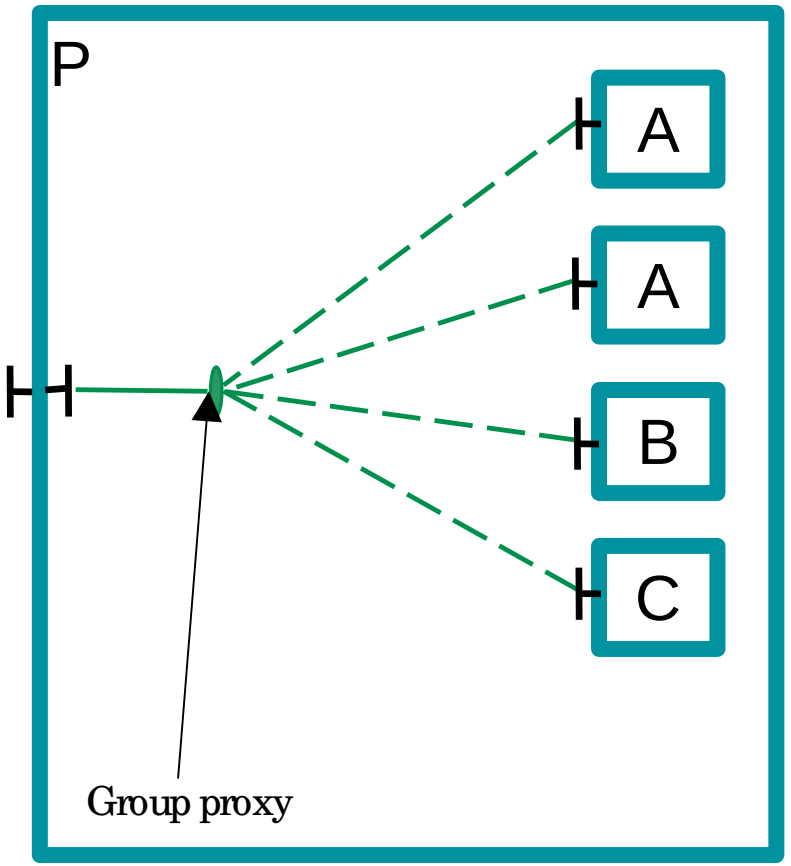
ObjectWeb
Open Source Middleware

Components based on Fractal Model

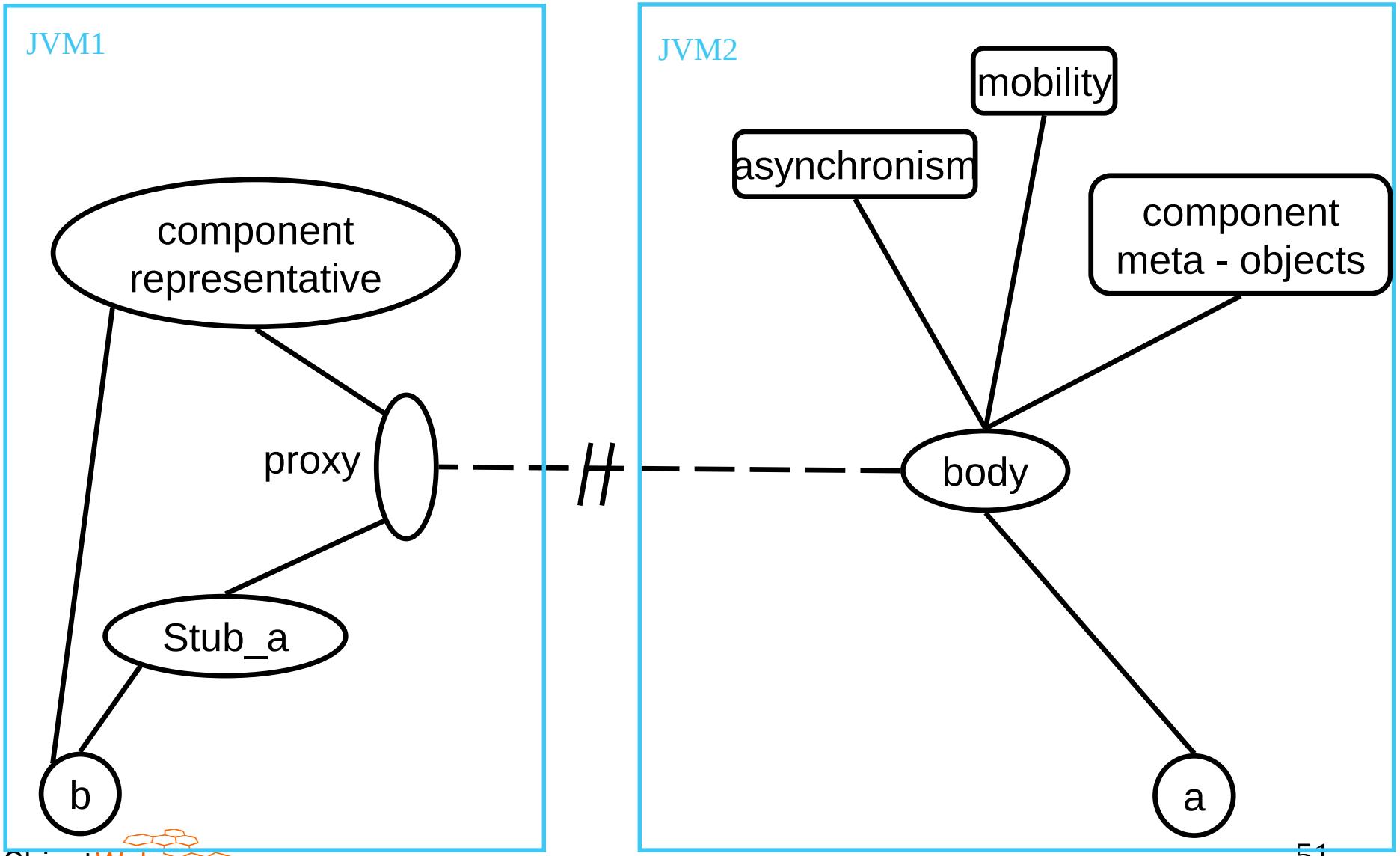


Distributed and Parallel Components

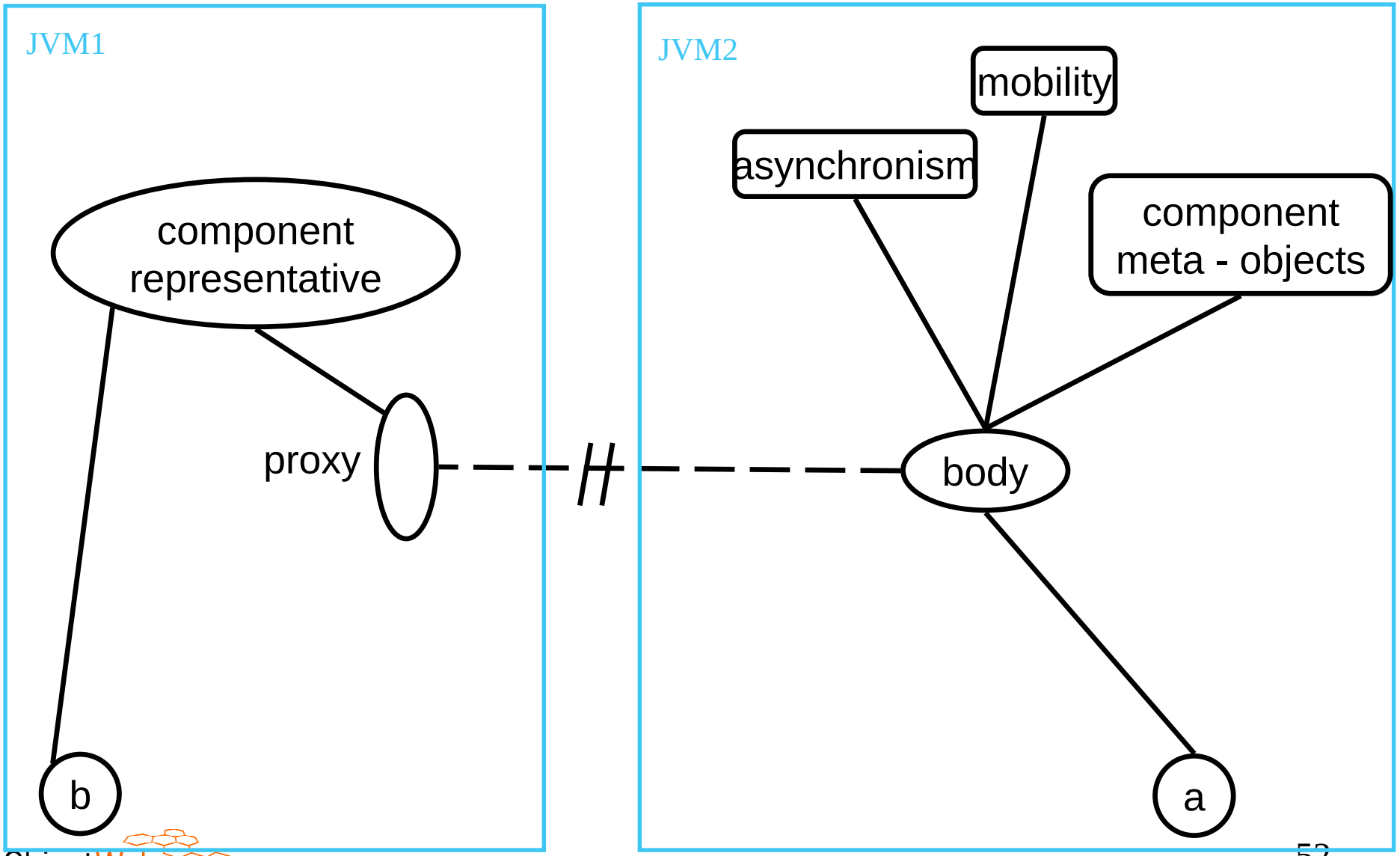




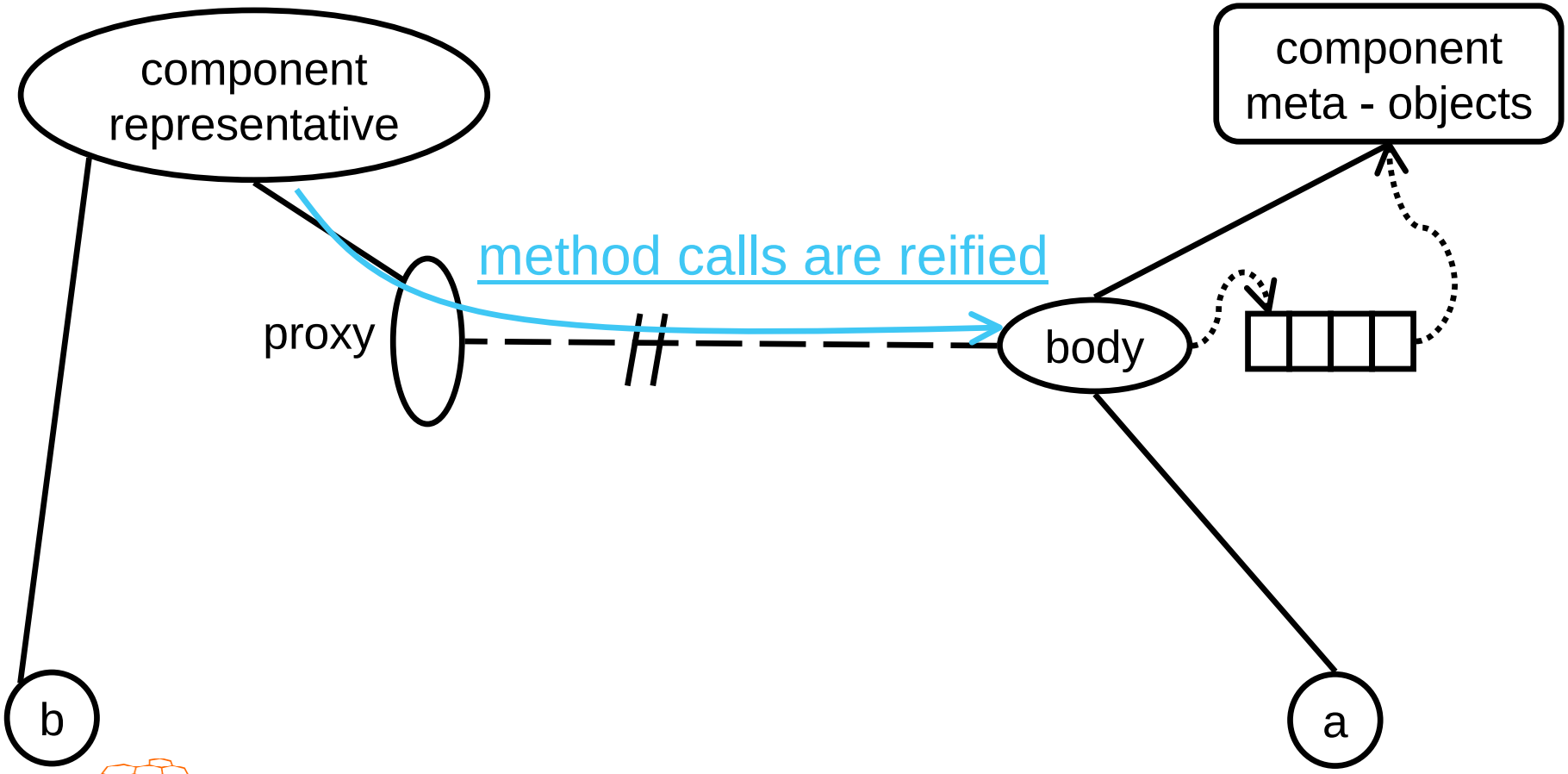
Components Infrastructure



Components Infrastructure



Components Request



ObjectWeb
Open Source Middleware

Table of Contents

ObjectWeb
Open Source Middleware

Security

- **Non-functionnal security**
  - located inside the meta-level, transparent for applications.
- **Hierarchical domains**
- **Dynamic policy negotiation**
- **Certification chain to identify users, JVMs, objects**
  - User certificate => Application certificate => active object certificate
  - User private key used only once for generating application certificate
- **Security policies set by deployment descriptors**

Request to an Active Object