ProActive : a tutorial

Matthieu Morel, Christian Delbe,
Virginie Legrand, Alexandre DiCostanzo

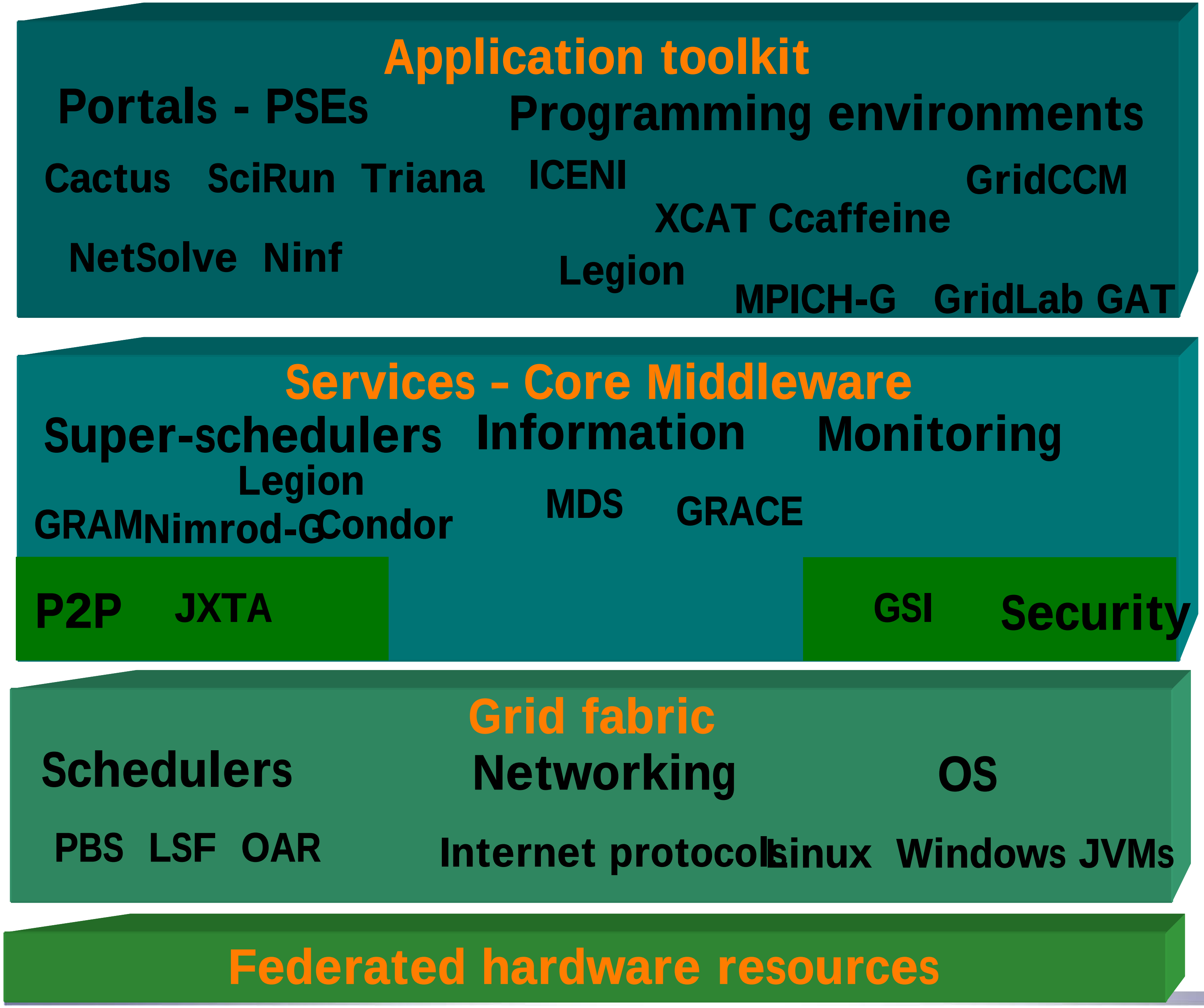
ETSI, Sophia-Antipolis
October 11th 2005

Agenda

- Objectives of the project
- Team
- Library
  - Features
  - Example : C3D + IC2D
  - Architecture
- An example based on a computation of π
  - Active objects
  - Groups
  - Deployment (LAN, P2P)
  - Web services
- A fault-tolerant deployment of a n-body application

Objectives

- Grid computing as a target
  - Programming model
    - ASP, asynchronism, groups, components
  - Programming environment
    - Library, monitoring tools, deployment framework
- Providing a support for research work
  - Formal models, proofs, model checking
  - Research on tools, models and protocols for Grid computing
- Building an international community
  - Feedback, requests for enhancements
  - Support
  - Gatherings



Application toolkit

Portals - PSEs

Programming environments

Cactus SciRun Triana ICENI GridCCM

XCAT Ccaffeine

NetSolve Ninf

Legion

MPICH-G GridLab GAT

Services - Core Middleware

Super-schedulers Information Monitoring

Legion

GRAM Nimrod-G Condor MDS GRACE

P2P JXTA

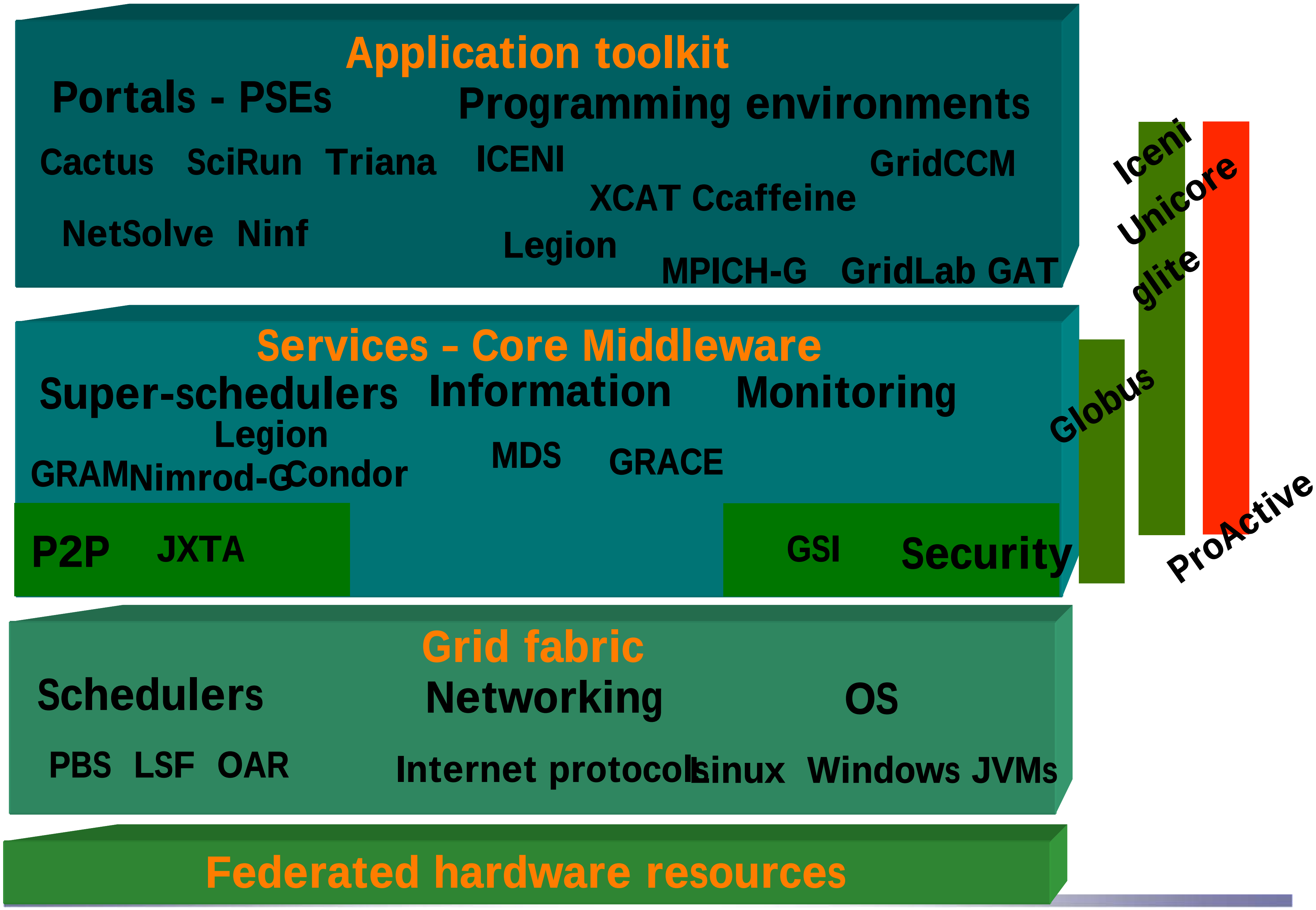
GSI Security

Grid fabric

Schedulers Networking OS

PBS LSF OAR Internet protocols Linux Windows JVMs

Federated hardware resources



Application toolkit

Portals - PSEs

Programming environments

Cactus SciRun Triana ICENI GridCCM

XCAT Ccaffeine

NetSolve Ninf

Legion

MPICH-G GridLab GAT

Services - Core Middleware

Super-schedulers Information Monitoring

Legion

GRAM Nimrod-G Condor

MDS GRACE

P2P JXTA

GSI Security

Grid fabric

Schedulers Networking OS

PBS LSF OAR

Internet protocols Linux Windows JVMs

Federated hardware resources

Iceni

Unicore

glite

Globus

ProActive

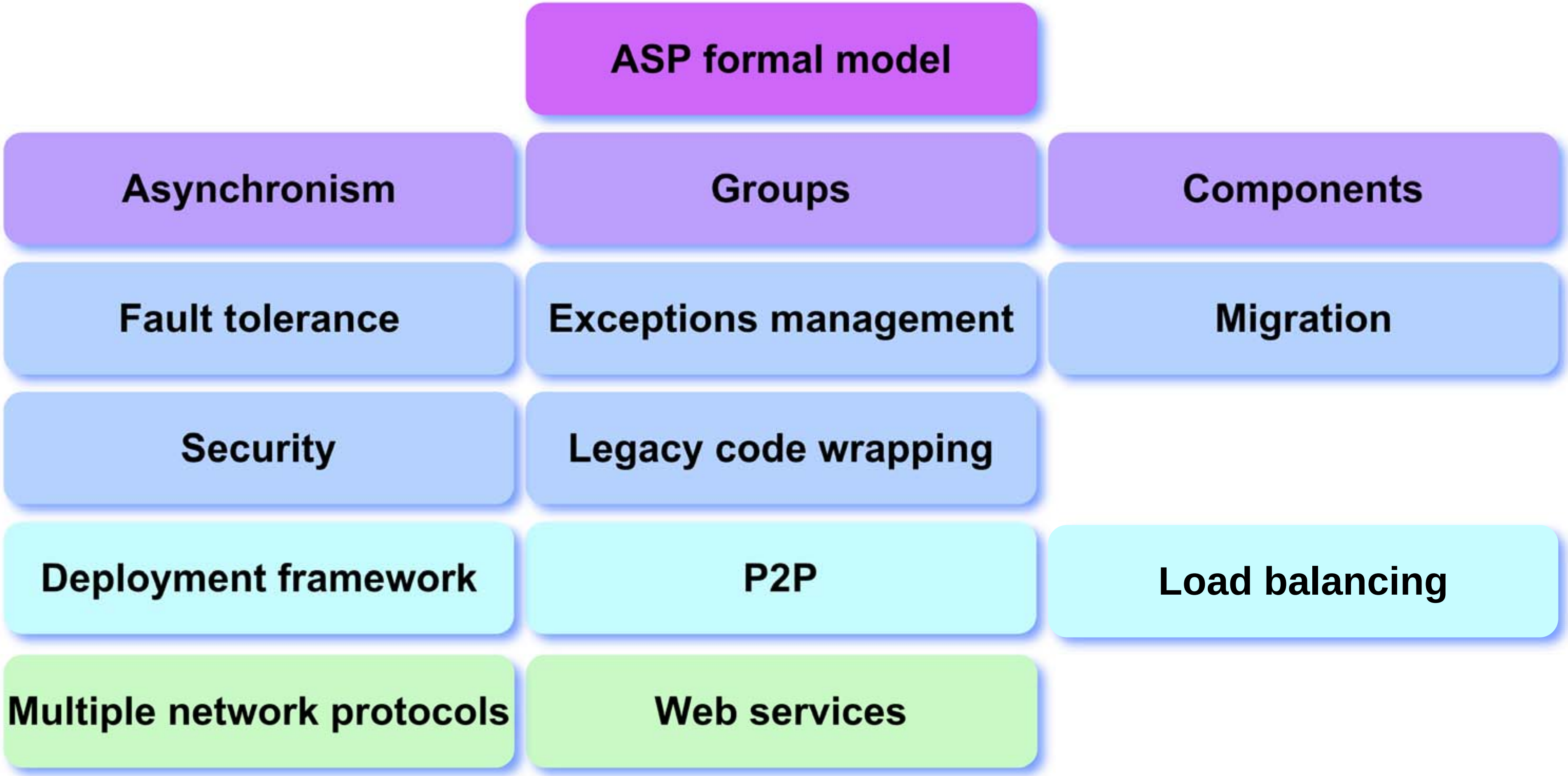
The team : 15+ members

- 3 professors
- 2 researchers
- 1 postdoc
- 4 engineers
- 5 PhD students
- + Interns…

The library

- Started in 1999
- Website with documentation
- Mailing list for support / questions / feedback
- Official releases ~ every 6 months
  - ProActive 3.0 scheduled for november
- Anonymous CVS access
- Metrics :
  - 1043 classes, ~ 63000 lines of code
  - 74 regression tests

Features



ASP formal model

Asynchronism

Groups

Components

Fault tolerance

Exceptions management

Migration

Security

Legacy code wrapping

Deployment framework

P2P

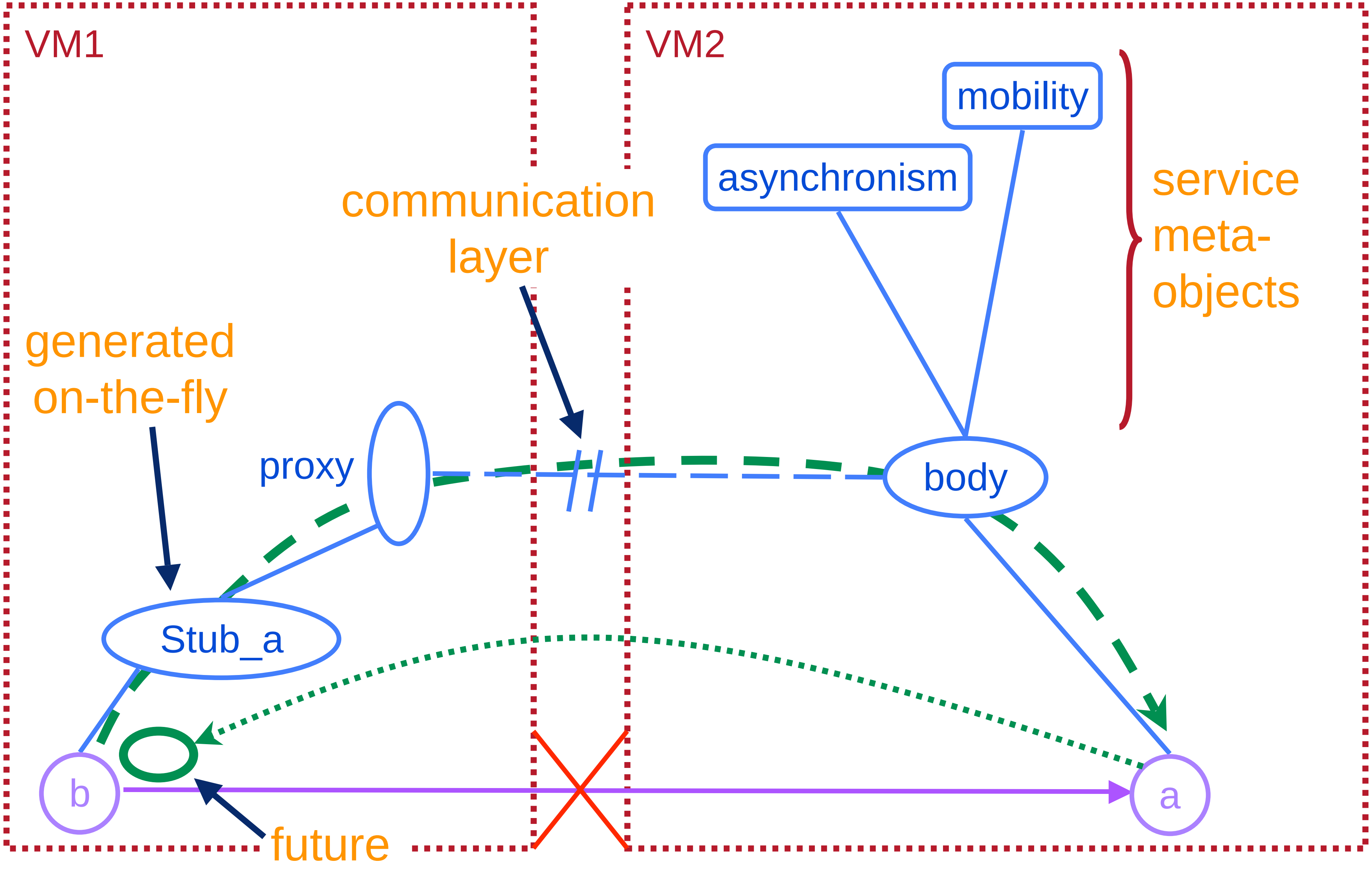
Load balancing

Multiple network protocols

Web services

Example : IC2D + C3D

Architecture : a Meta-Object Protocol



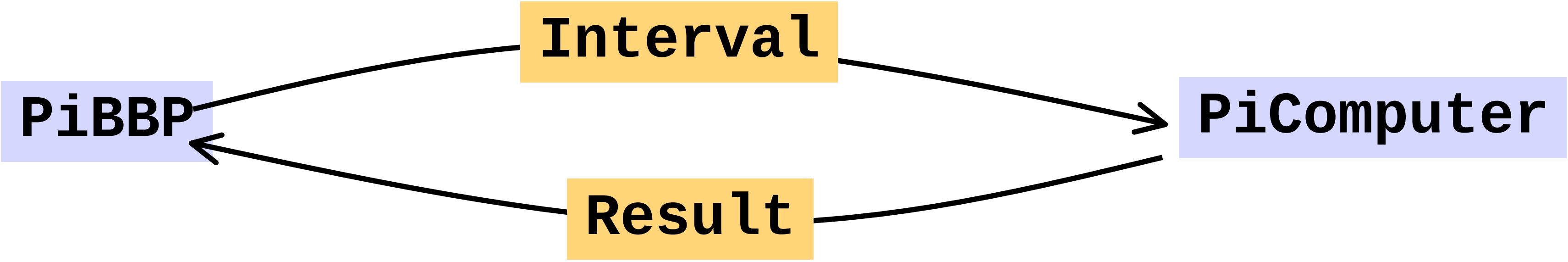
Programming and deploying

A computation of π

- An evaluation of π using the Bailey-Borwein-Plouffe formula

$$\boxed{\pi = \sum_{n=0}^{\infty}\left(\frac{4}{8n+1}-\frac{2}{8n+4}-\frac{1}{8n+5}-\frac{1}{8n+6}\right)\cdot\left(\frac{1}{16}\right)^{n}}$$

- 4 classes



PiBBP

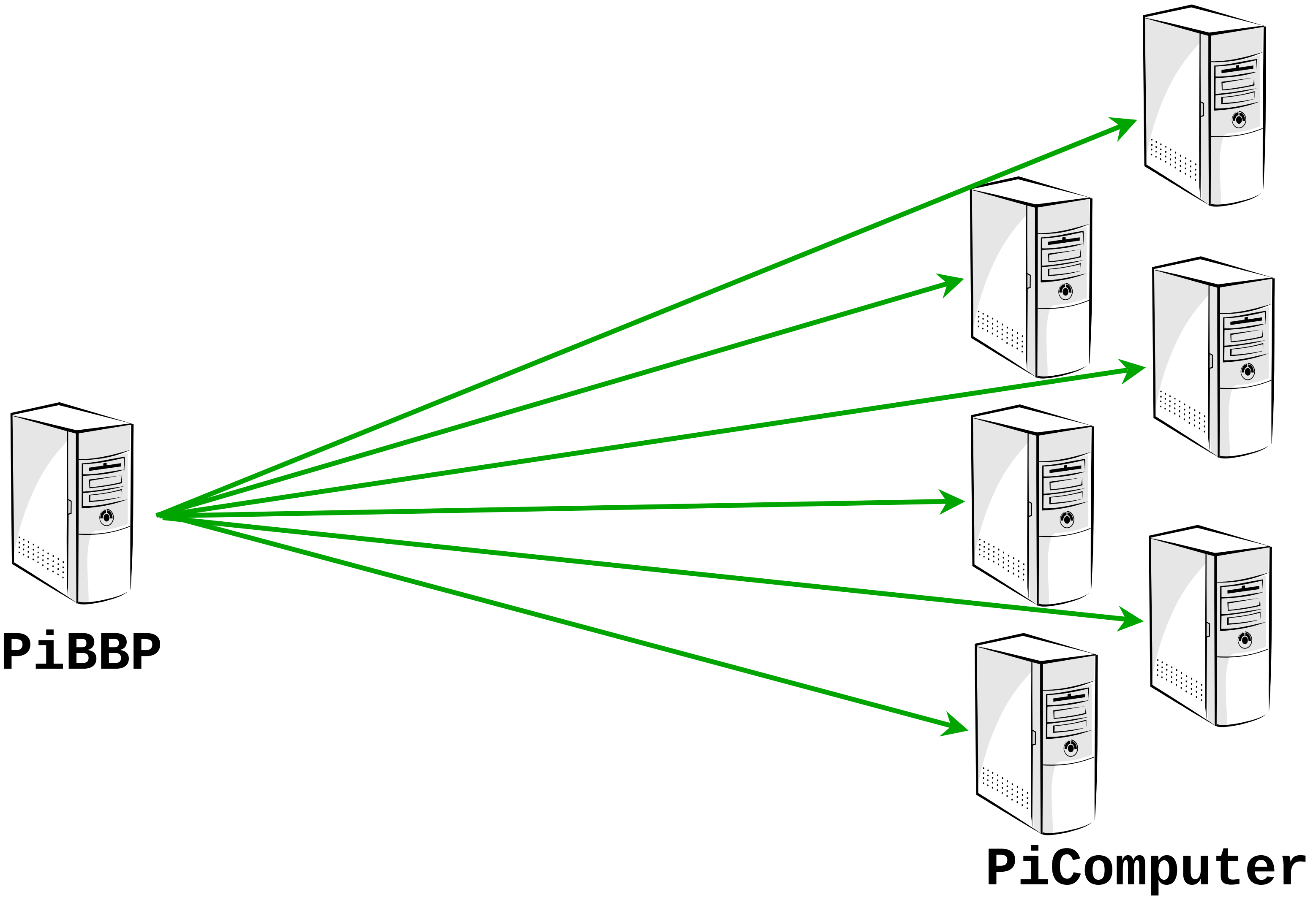
Interval

PiComputer

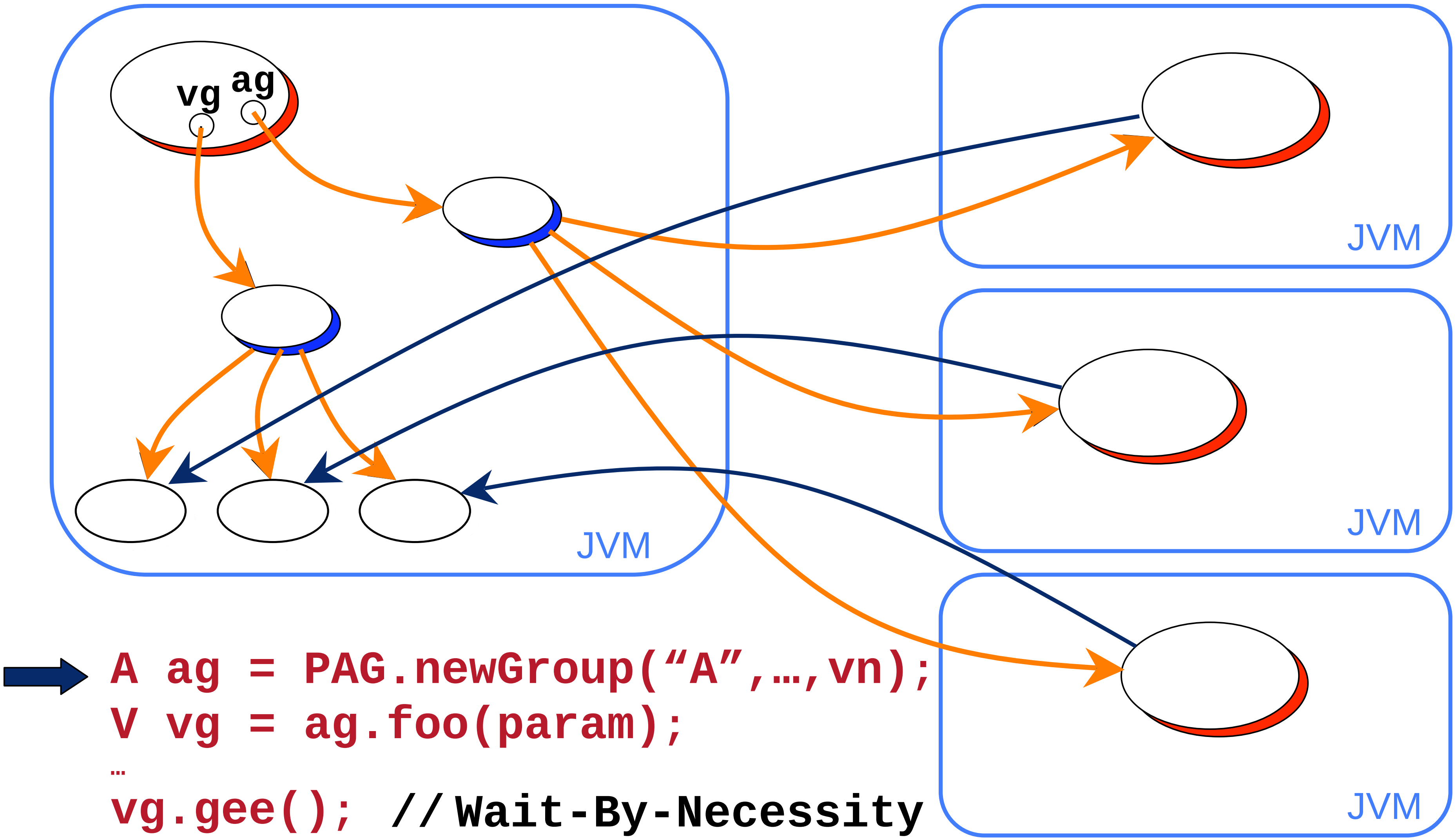
Result

Distributing the computation

- An algorithm suited for distribution (master-slaves)

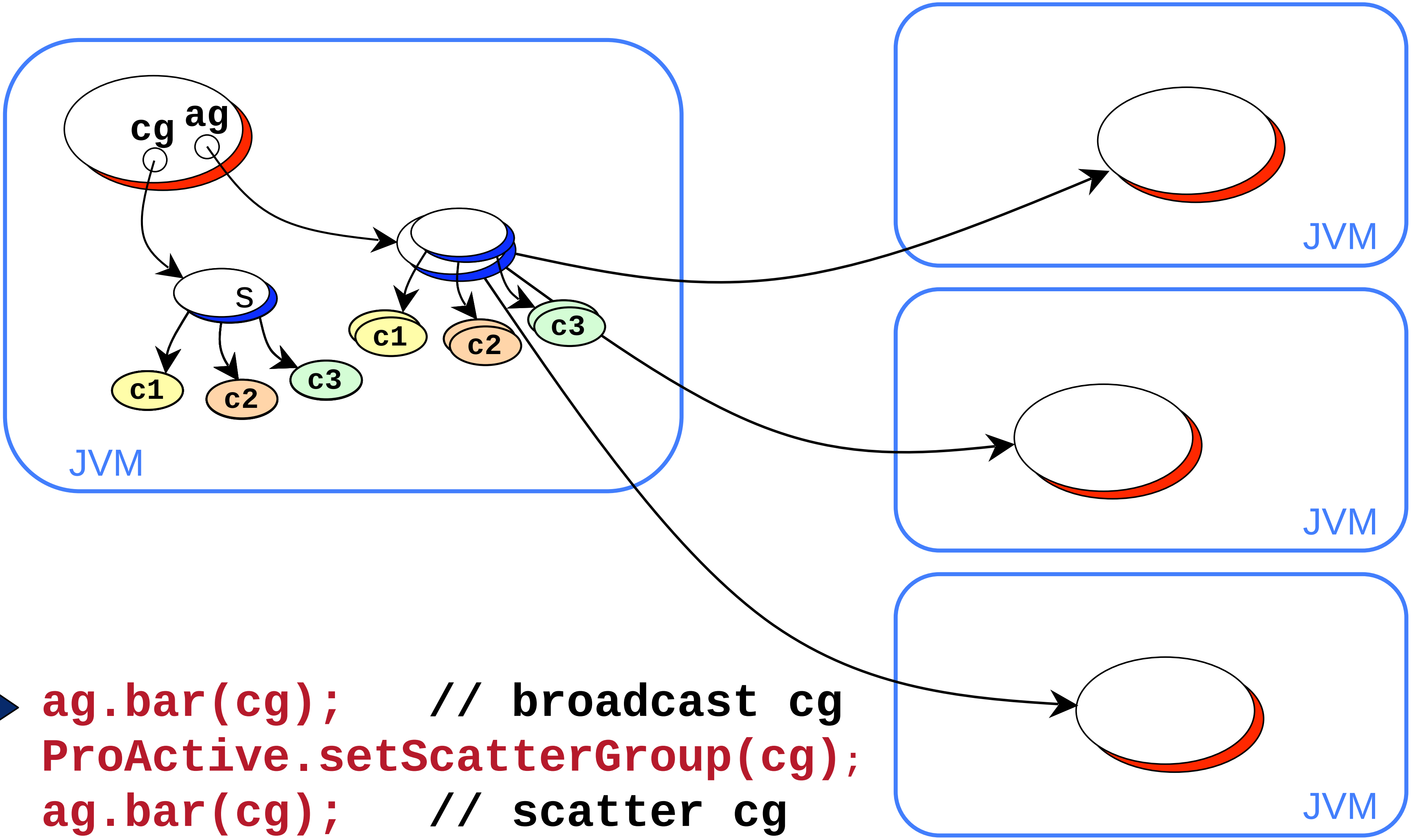


Group call



```
A ag = PAG.newGroup(“A”,…,vn);
V vg = ag.foo(param);
…
vg.gee(); // Wait-By-Necessity
…
```

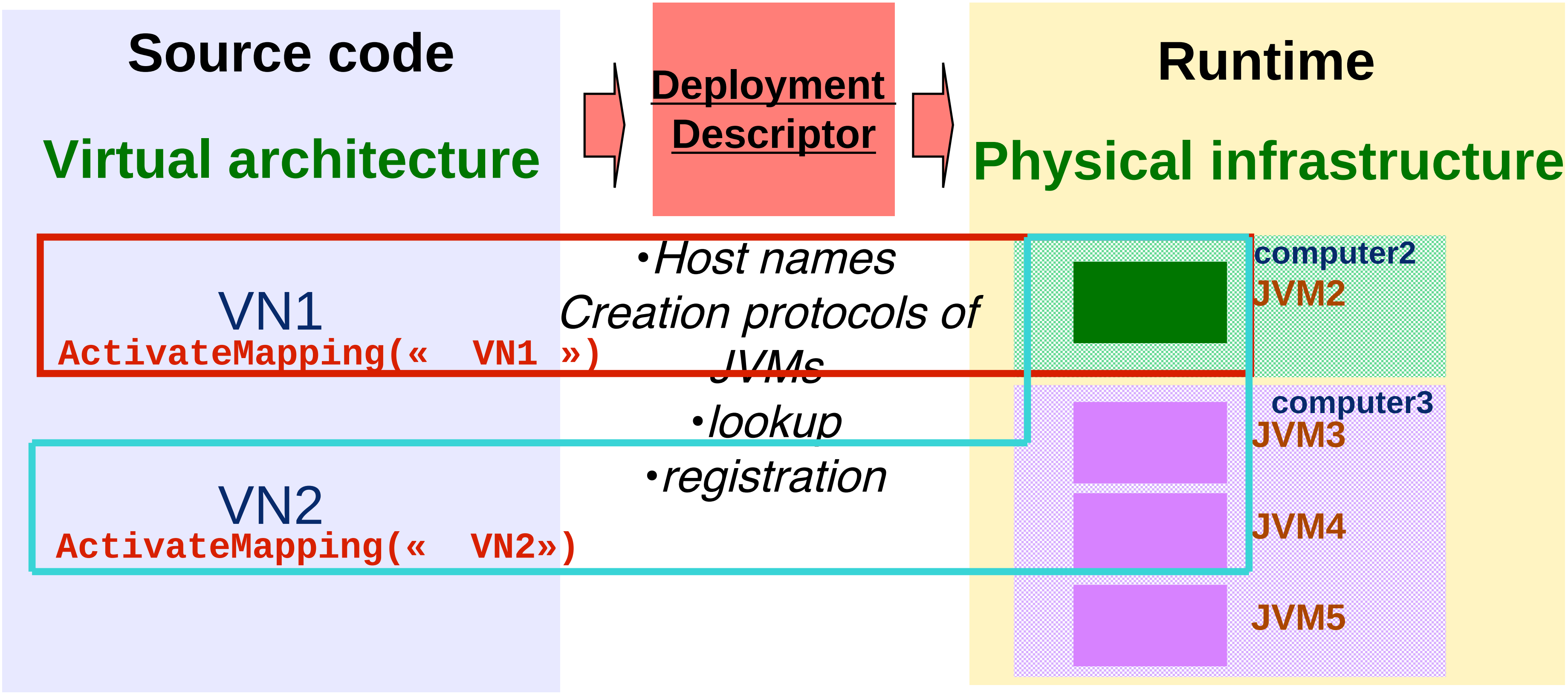
Broadcast or scatter



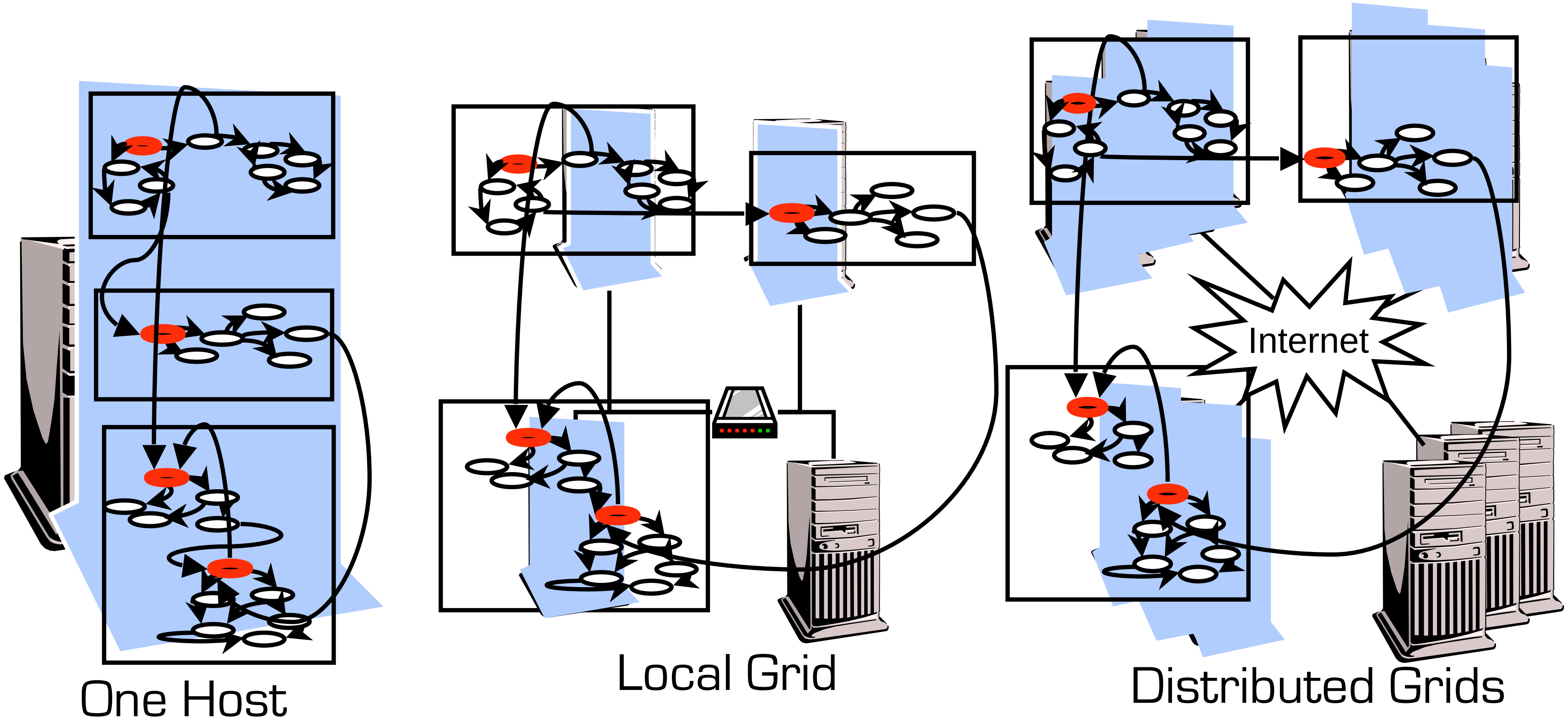
```
ag.bar(cg);   // broadcast cg
ProActive.setScatterGroup(cg);
ag.bar(cg);   // scatter cg
```

Abstract deployment model

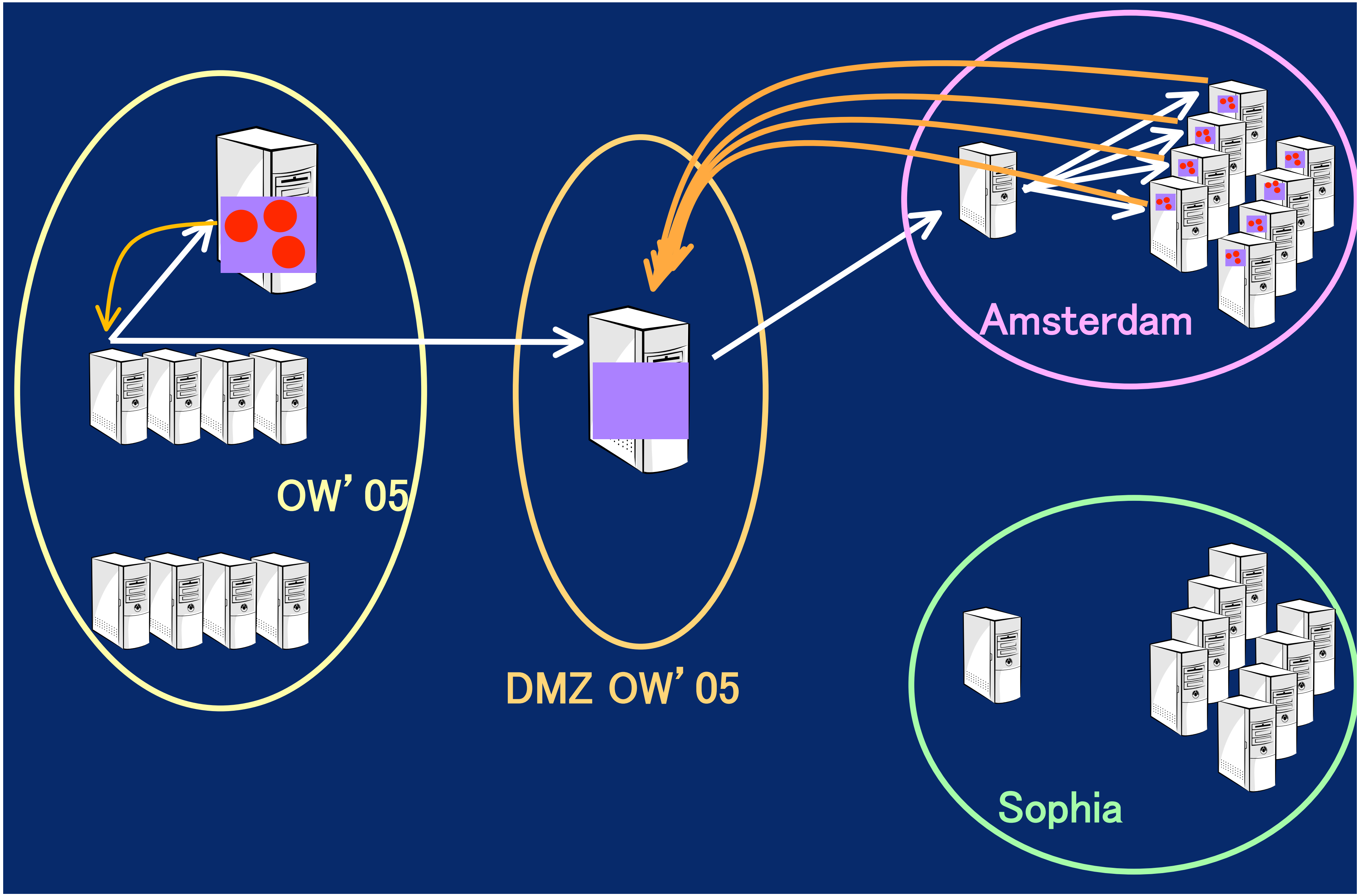
- Separates design from deployment infrastructure
- Virtual nodes
- Dynamic enactement of the deployment, from the application



Same application, many deployments



A simple Grid set-up



Components : Rationale

- Observation : complexity and heterogeneity of the Grid
  - ➡ complexity (design, deployment and reusability)
  - ➡ performance issues

- Answer : framework for programming and deploying components on the Grid
  - ➡ implementation of the Fractal model for ProActive
  - ➡ extensions for the Grid

Objective : a framework for Grid components

- Facilitating the design and implementation of complex distributed systems

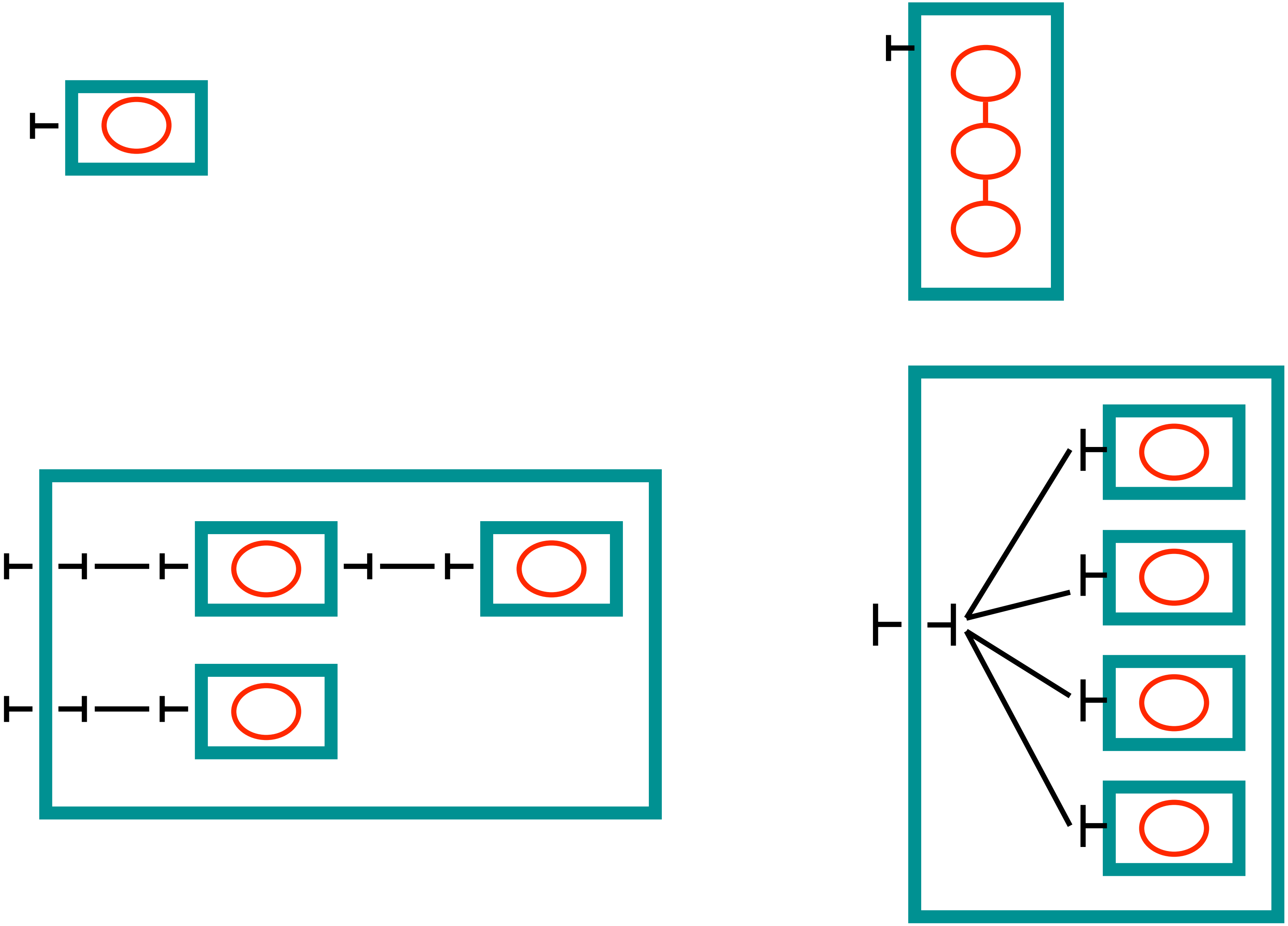
- Leveraging the ProActive library

  ProActive components benefit from underlying features

- Allowing reuse of legacy components (e.g. MPI)

- Providing tools for defining, assembling and monitoring distributed components

ProActive components : 4 flavors



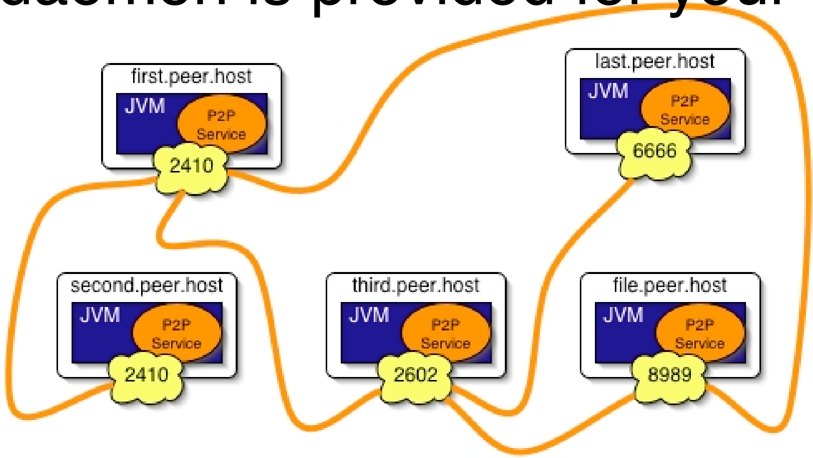
Peer-to-Peer computing

Alexandre di Costanzo

Peer-to-Peer Infrastructure

- It is not a big deal, just modify the XML Descriptor
- The only difference with a classic ProActive deployment is that it does not create JVM:

  JVMs are already running on remote hosts
- You will acquire computational nodes from a P2P JVM sharing infrastructure
- A nice option is you can ask for MAXimun nodes
- A GNU/Linux daemon is provided for your machines



Using P2P for deploying

```
<jvm name="p2pJvm">
        <acquisition>
                <serviceReference refid="p2pservice"/>
        </acquisition>
</jvm>
```

```
<infrastructure>
        <services>
                <serviceDefinition id="p2pservice">
                        <P2PService nodesAsked="10">
                                <peerSet>
                                        <peer>rmi://apple</peer>
                                        <peer>rmi://tranquility</peer>
                                        <peer>rmi://amstel</peer>
                                </peerSet>
                        </P2PService>
                </serviceDefinition>
        </services>
</infrastructure>
```

A Short Demo with INRIA P2P Desktop Grid

- About 40 machines the day
- About 200 machines the night

- Machines are using by their owners at this same moment

ProActive interoperability with Web Services

Virginie LEGRAND

Virginie.Legrand@sophia.inria.fr

Expose Active Object and components interfaces as Web Services

Mechanism

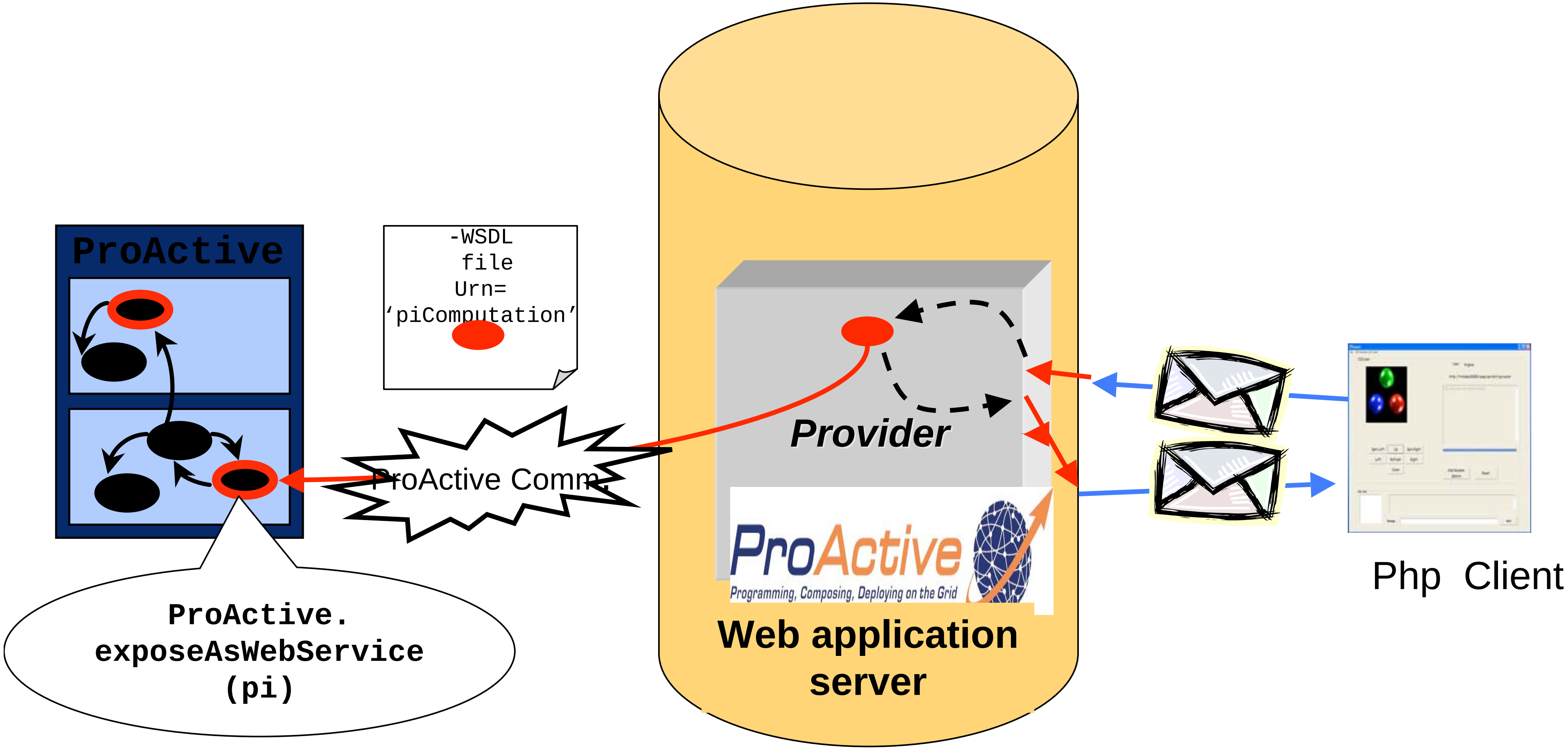
Web Service Integration

- **Aim:**
  - Turn active objects and components interfaces into Web Services

    ➔ interoperability with any foreign language or any foreign technology.

- **API**
  - Expose an **active object** as a **web Service** on a web server :
    - the user can choose the methods he wants to expose.
    - **`exposeAsWebService`**
      `(Object o, String url, String urn, String [] methods );`
  - Expose the **interfaces of a component** as web services :
    - **`exposeComponentAsWebService`** `(Component component, String url, String componentName );`

WSDL file available at :
http://localhost:8080/soap/servlet/wsdl?id=piComputation



3. Client Call

2. Deploy/exposeAsWebService ()

Fault tolerance

Christian Delbé

Fault-tolerance in ProActive

- Rollback-Recovery fault-tolerance
  - After a failure, revert the system state back to some earlier and correct version
  - Based on periodical checkpoints of the active objects
  - Stored on a stable server

- Two protocols are implemented
  - Communication Induced Checkpointing (CIC)
    - + Low failure free overhead
    - – Slow recovery
  - Pessimistic Message Logging (PML)
    - – Higher failure free overhead
    - + Fast recovery

- Transparent and non intrusive

Fault-tolerance Server

- A global server is implemented
  - Checkpoint storage
  - Failure detection
    - Detect fail-stop failures
  - Localization service
    - Return the new location of a failed object
  - Resource service
    - Return a host node for recovering a failed object
    - Based on deployment or on underlying P2P infrastructure

```
~/ProActive/scripts/unix/FT> ./startGlobalFTServer.sh
  [-proto cic|pml]
  [-name name]
  [-port portnumber]
  [-fdperiod faultDetectionPeriod (sec)]
  [-p2p serverUrl]
```

Fault-tolerant deployment

- Fault-tolerance is set in deployment descriptors
  - Fault-tolerance service attached to virtual nodes
  - No source code alteration

```
<virtualNode name="Workers" ftServiceId="ft"/>
...
<serviceDefinition id="ft">
  <faultTolerance>
   <protocol type="cic"/>
   <globalServer url="rmi://host/FTServer"/>
   <resourceServer url="rmi://host/FTServer"/>
   <ttc value="10"/>
  </faultTolerance>
</serviceDefinition>
```

Fault-tolerant deployment

- Fault-tolerance is set in deployment descriptors
  - Fault-tolerance service attached to virtual nodes
  - No source code alteration
- Protocol selection

```
<virtualNode name="Workers" ftServiceId="ft"/>
...
<serviceDefinition id="ft">
  <faultTolerance>
   <protocol type="cic"/>
   <globalServer url="rmi://host/FTServer"/>
   <resourceServer url="rmi://host/FTServer"/>
   <ttc value="10"/>
  </faultTolerance>
</serviceDefinition>
```

Fault-tolerant deployment

- Fault-tolerance is set in deployment descriptors
  - Fault-tolerance service attached to virtual nodes
  - No source code alteration
- Protocol selection
- Server(s) location

```
<virtualNode name="Workers" ftServiceId="ft"/>
...
<serviceDefinition id="ft">
  <faultTolerance>
   <protocol type="cic"/>
   <globalServer url="rmi://host/FTServer"/>
   <resourceServer url="rmi://host/FTServer"/>
   <ttc value="10"/>
  </faultTolerance>
</serviceDefinition>
```

Fault-tolerant deployment

- Fault-tolerance is set in deployment descriptors
  - Fault-tolerance service attached to virtual nodes
  - No source code alteration
- Protocol selection
- Server(s) location
- Checkpoint period

```
<virtualNode name="Workers" ftServiceId="ft"/>
...
<serviceDefinition id="ft">
  <faultTolerance>
   <protocol type="cic"/>
   <globalServer url="rmi://host/FTServer"/>
   <resourceServer url="rmi://host/FTServer"/>
   <ttc value="10"/>
  </faultTolerance>
</serviceDefinition>
```

Demo : N-Body application

- Bodies in movement under gravitational force
  - Example available in ProActive release
- SPMD application
  - Fully connected
- Naïve algorithm
  - One iteration = n*(1 to n) communications
- Demo
  - Classic deployment
  - Fault-tolerance deployment with CIC protocol
  - Fault-tolerance deployment with PML protocol