Dynamically-Fulfilled Application Constraints through Technical Services

Towards Flexible Component Deployments

Denis Caromel, Christian Delbé,
Alexandre di Costanzo, and Matthieu Morel

HPC GECO / Compframe, Paris France
Tuesday 20th June 2006

OASIS Research Group
INRIA Sophia - I3S - CNRS - University of Nice Sophia Antipolis France

Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

Constrained Deployment

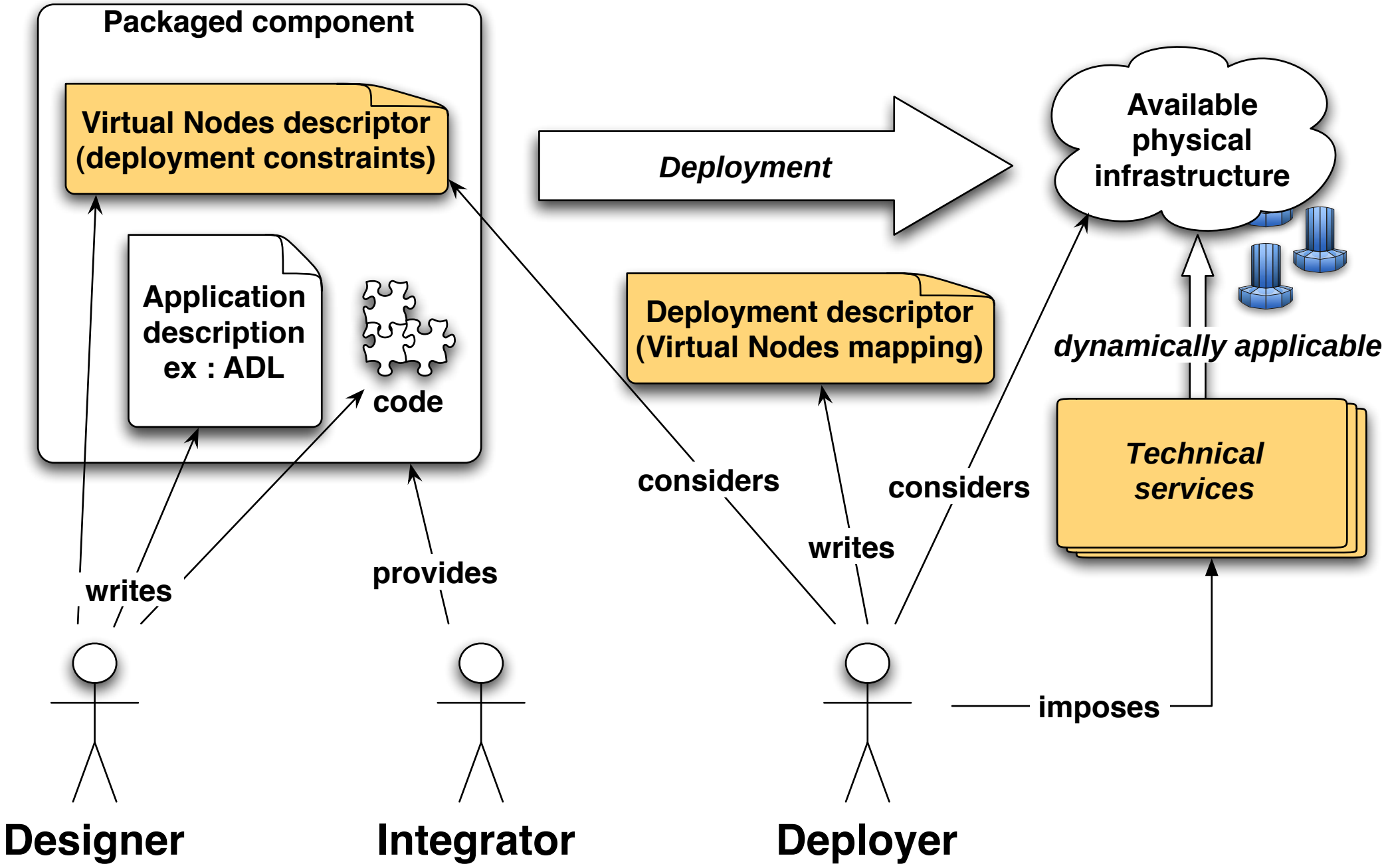
- Components may require **non-functional** services:
  - security, fault-tolerance, etc.
- Some constraints may express **deployment requirements**:
  - number of resources expected, timeout, etc.

Describe non-functional and deployment requirements as contract in a descriptor of Virtual Nodes

Constraints

- Strict **separation** of non-functional requirements from the code
  - Using: **descriptor of Virtual Nodes**
- 2 kinds of constraints:
  - **Statically** fulfilled requirements
    - Example: Operating Systems
  - **Dynamically** fulfilled requirements
    - Example: fault-tolerance

Deployment Roles and Artifacts

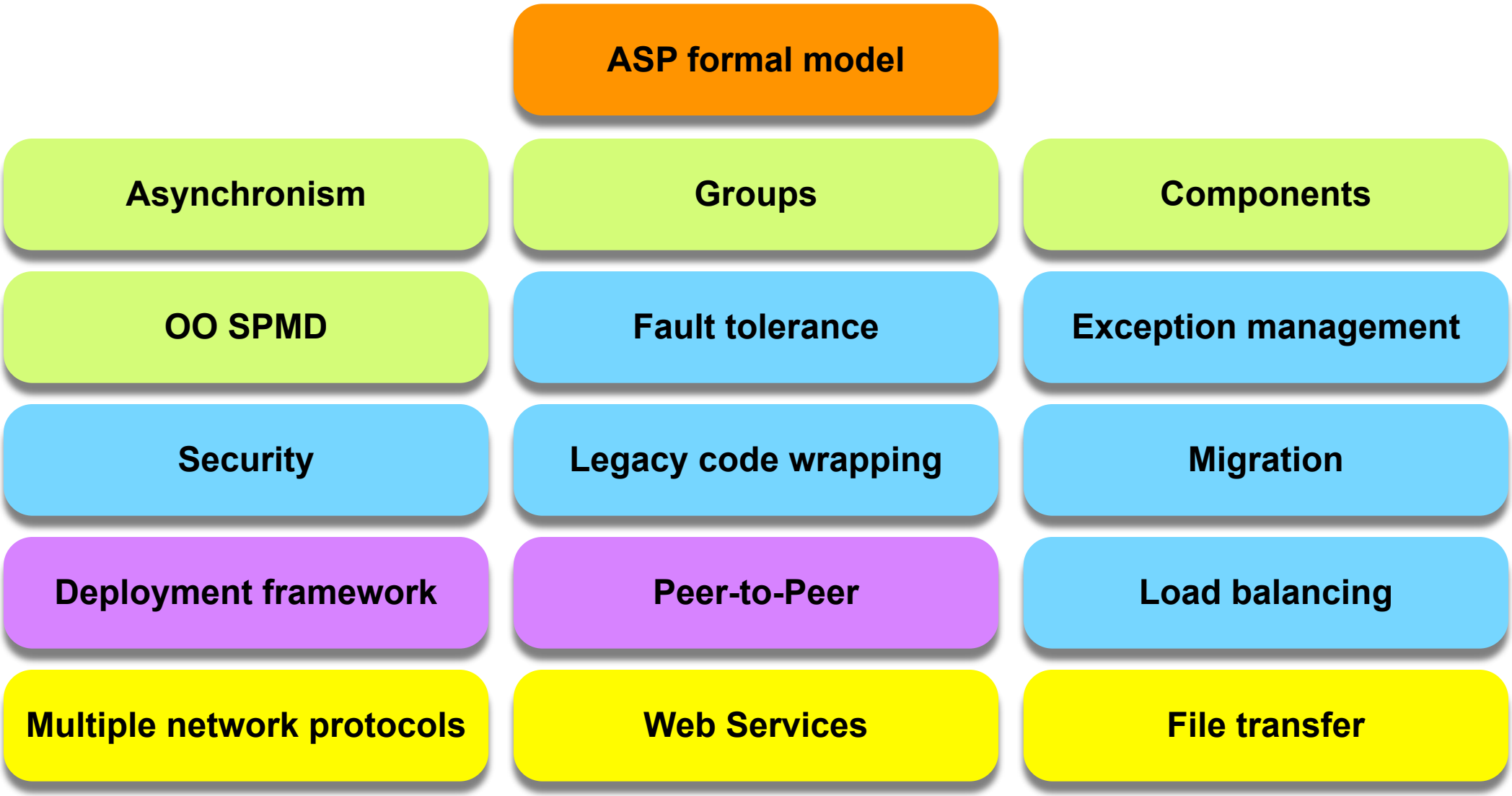


Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

ProActive Middleware

A Java API + Tools for Parallel and Distributed Computing



Programing - Wrapping - Composing - Deploying

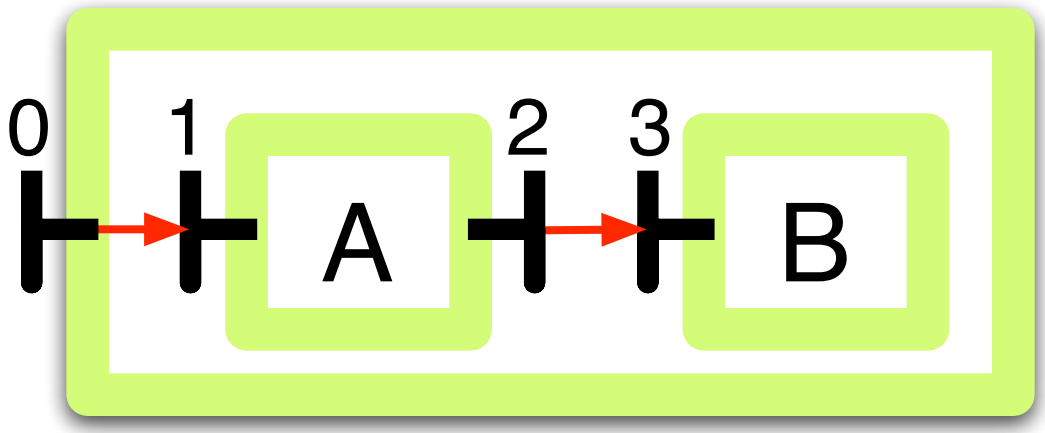
Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

Component Based Programming

- Implementation of the **Fractal model** with ProActive:
  - Components are implemented as active objects
  - **Hierarchical** components
  - **Distributed** components
- Deployment of components:
  - Standardized Architecture Description Language (**ADL**)

ADL Example



ADL

```
Composite
  interface server 0
  component A
     interface server 1
     interface client 2
     vn name=”vn1”
  component B
     interface server 3
     vn name = “vn2”
  Bindings
     this.0 → A.1
     A.2    → B.3
```

Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

Deployment Framework

- Avoid deployment specific source code;
- Avoid scripting for configuration, getting nodes, connecting, etc.
- Abstract from source code:
  - Machine names
  - Creation protocols
  - Lookup and registry protocols

XML Deployment file ⇔ Virtual Node (VN) ⇔ Application

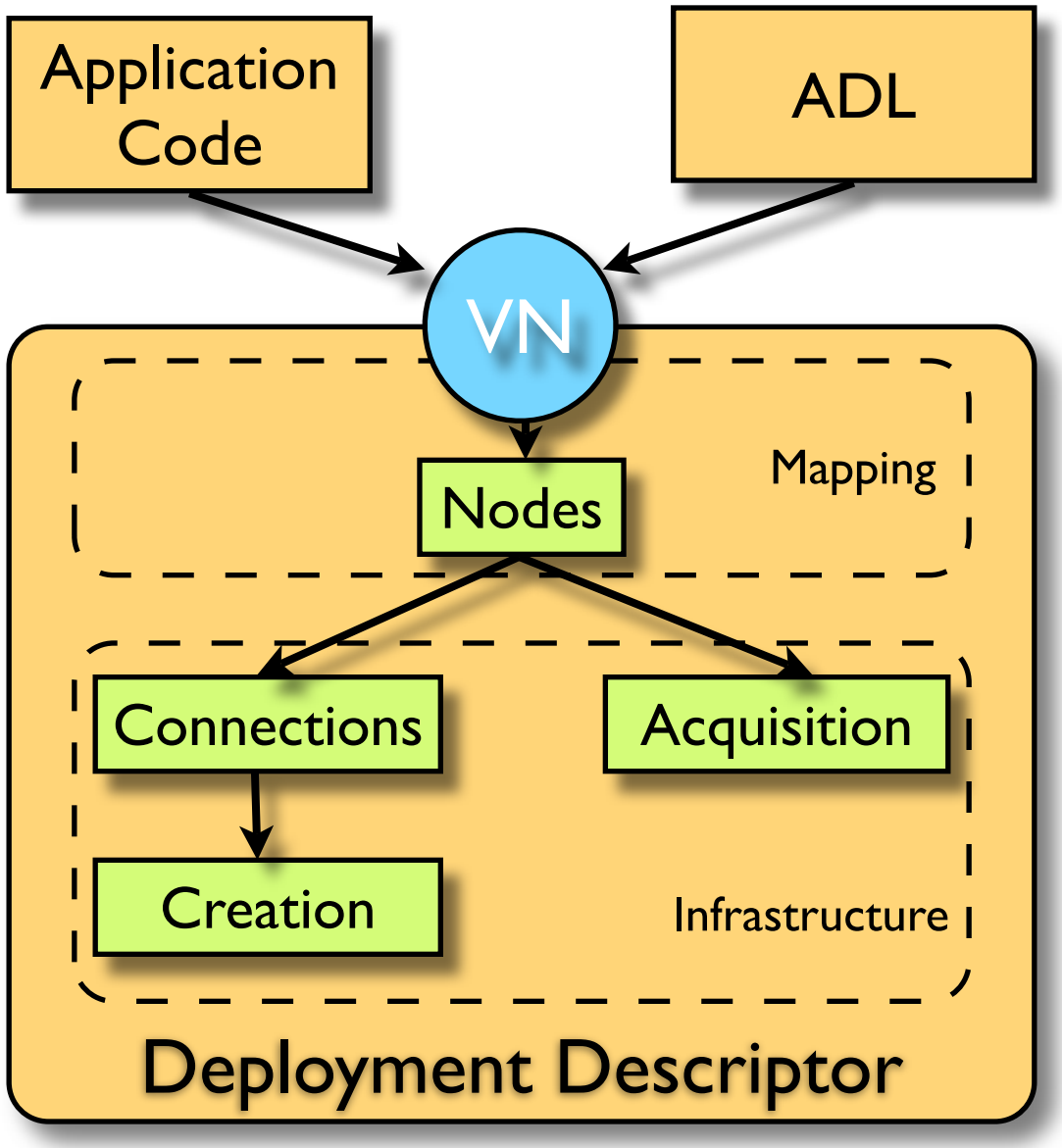
Supported protocols:
gLite, Unicore, Sun Grid Engine, Globus , ssh, rsh, LSF, PBS, etc.

Virtual Node (VN)

- Identified as a string name
- Used in program code source and/or ADL
- Configured in an XML descriptor file
- Node:
  - ProActive execution environment
  - Mapping of VN to Nodes and to JVMs

Program Source	Descriptor
Activities → **VN**	VN → **Nodes** → JVMs

Deployment Descriptor



Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

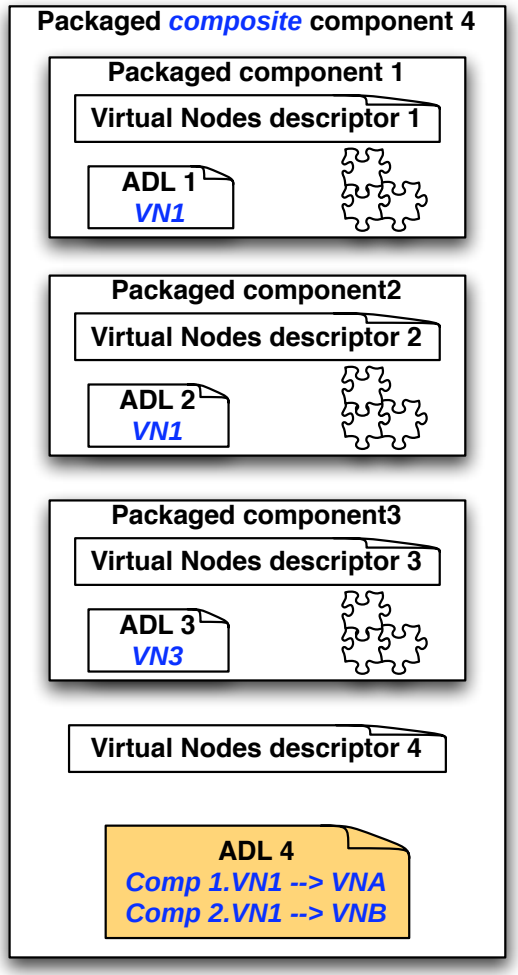
Technical Services

- A **non-functional requirement** that may be dynamically fulfilled by adapting the configuration of selected resources
- In Deployment Descriptor:
  - **define configuration** in a technical service
  - **Apply a technical service on a virtual node**
- Virtual Node abstracts the nature of nodes
  - The configuration is similarly applied on a **created and acquired node**
- From the technical service programmer point of view:
  - Interface: TechnicalService
- From the deployer point of view:
  - Set of "variable-value" tuples

Virtual Node Descriptors

```
<virtual-nodes>
   <virtual-node name="VN1">
      <technical-service
      type="services.FaultTolerance"/>
      <processor architecture="x86"/>
      <os name="linux" release="2.6.15"/>
   </virtual-node>
</virtual-nodes>
```

Composition of Components with renaming of Virtual Nodes




```
<virtual-nodes>

    <virtual-node name="VNA">
        <technical-service type="services.FT"/>
        <os name="linux" release="2.6.15"/>
    </virtual-node>

    <virtual-node name="VNB">
        <technical-service type="services.LB"/>
    </virtual-node>

    <virtual-node name="VN3" />

</virtual-nodes>
```

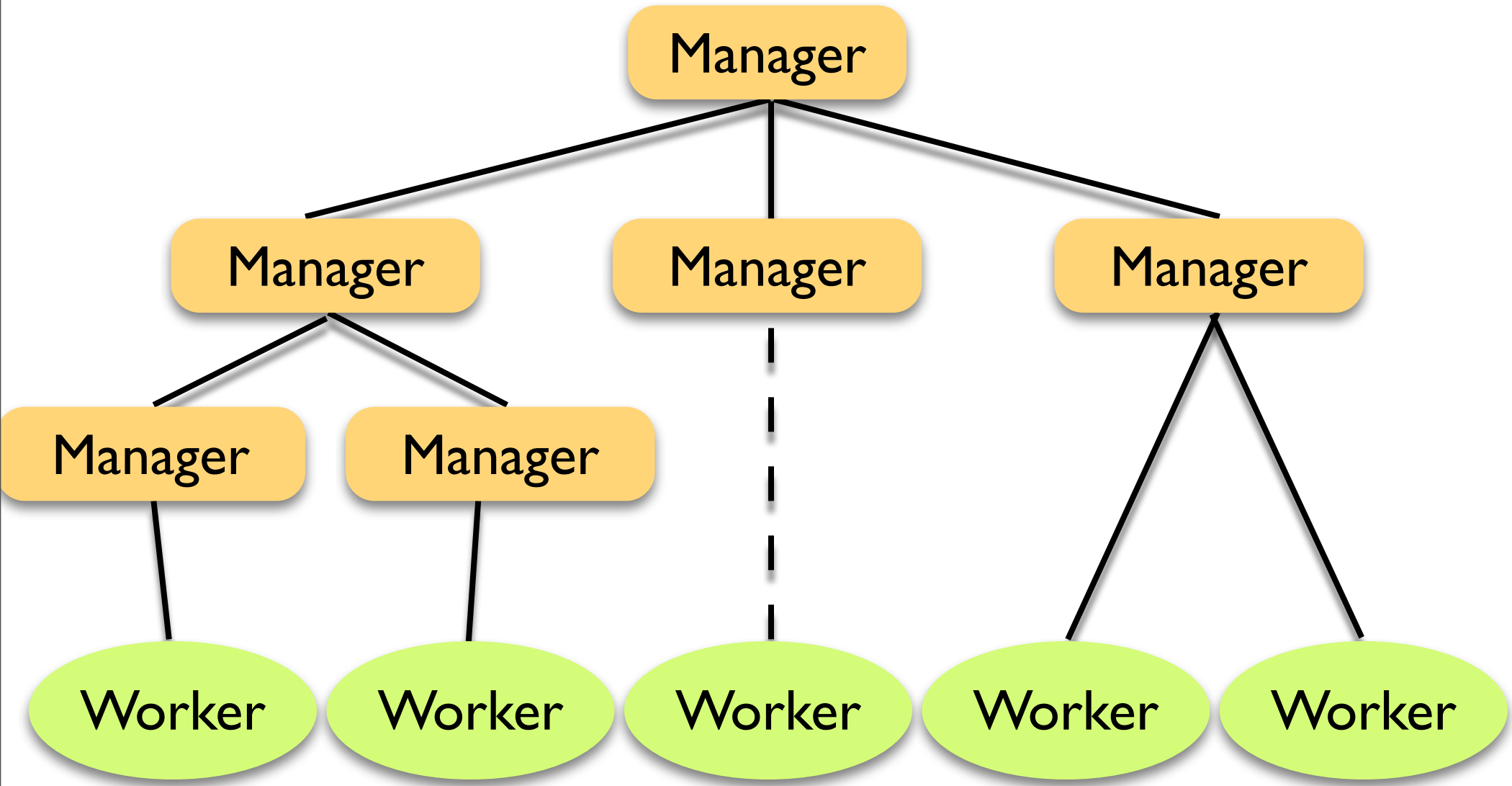
Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

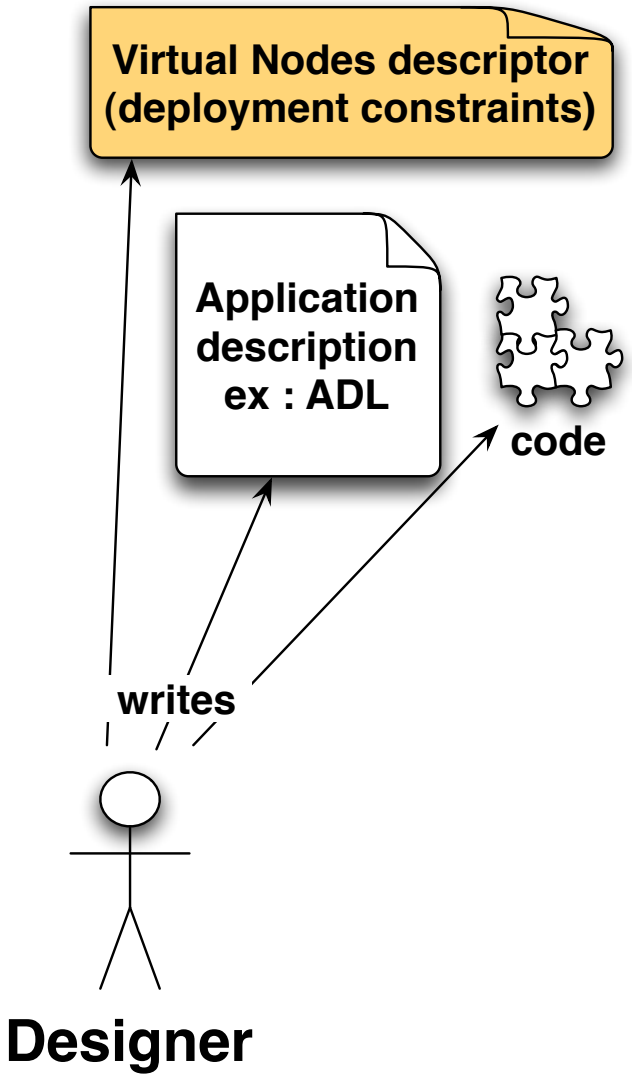
Deployment on a Peer-to-Peer Infrastructure with Fault-Tolerance requirements

- Fault-tolerance in ProActive:
  - Rollback-recovery: checkpoint-based
    - Communication-Induced Checkpointing (CIC)
    - Pessimistic Message Logging (PML)
    - ...
  - Defined in descriptors, not in source code: **Technical Service**
- P2P in ProActive: unstructured network for sharing JVMs

Flow-Shop: Master-Slaves



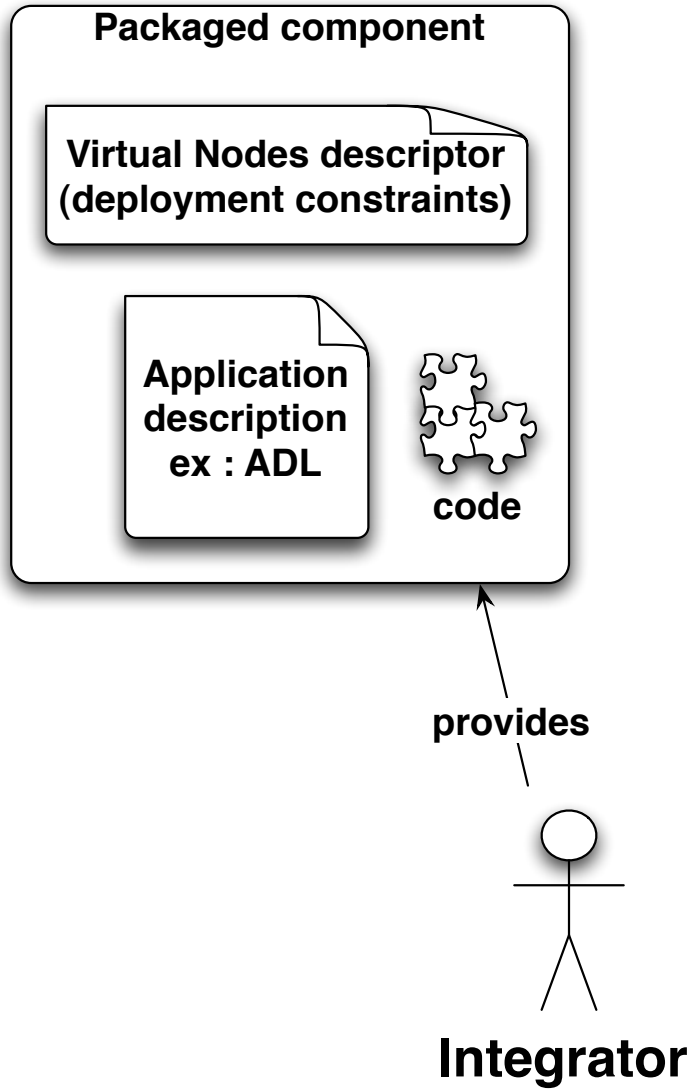
Deployment Roles and Artifacts



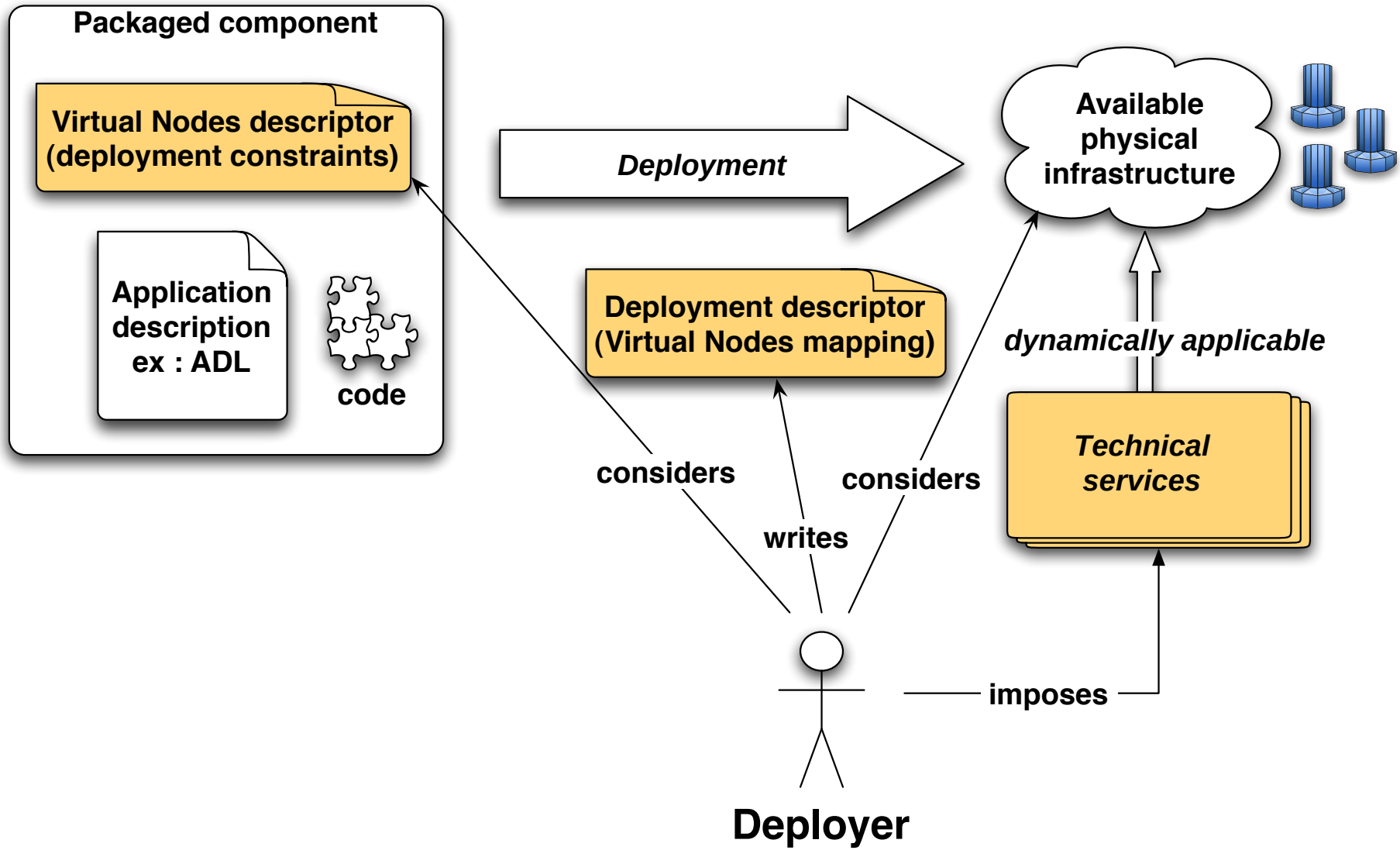
Virtual Nodes Descriptor

```
<virtual-nodes>
   <virtual-node name="managers">
      <technical-service
                  type="services.FaultTolerance"/>
      <processor architecture="x86"/>
   </virtual-node>
   <virtual-node name="workers"/>
</virtual-nodes>
```

Deployment Roles and Artifacts



Deployment Roles and Artifacts



XML Deployment Descriptor

```
<ProActiveDescriptor>
 <virtualNodeDefinition>
    <virtualNode name="managers" serviceRefid="ft-manager"/>
    <virtualNode name="workers"/>
 </virtualNodeDefinition>
 ...
 <technicalServiceDefinitions>
    <service id="ft-manager" class="services.FaultTolerance">
       <arg name="protocol" value="PML"/>
       <arg name="server" value="rmi://hostname/FTServer"/>
    </service>
 </technicalServiceDefinitions>
</ProActiveDescriptor>
```

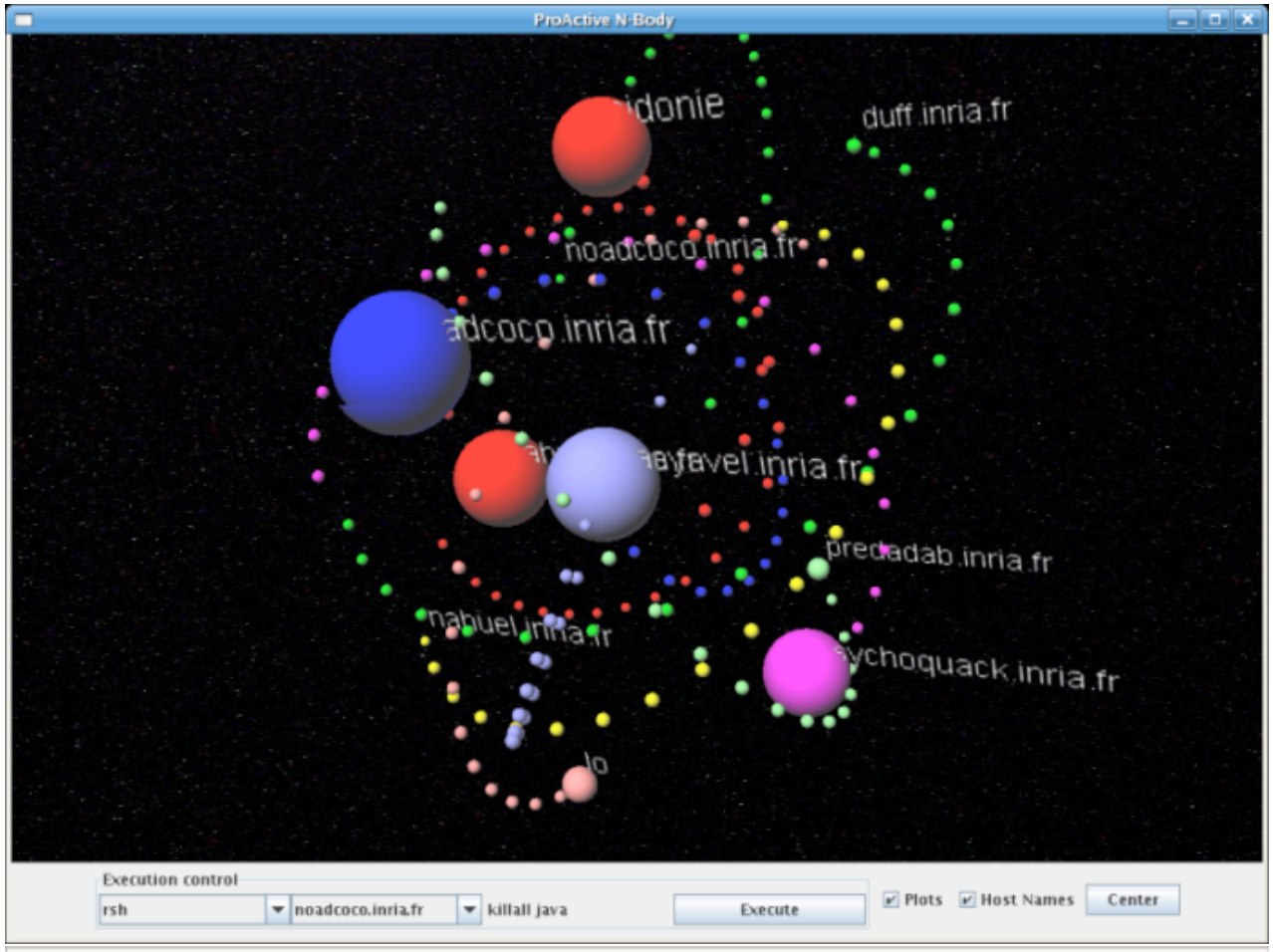
Agenda

- Problematic
  - Constrained deployment
- Context
  - The ProActive Grid Middleware
  - Component based programming
  - Deployment framework
- Solution
  - Virtual Node descriptor
- Use case: Fault tolerance & Peer-to-Peer
- Conclusion

Conclusion

- Implementation Status:
  - Technical Services implemented: Fault tolerance & load balancing
    - Problem for combining Technical Services:
      - Combining at the code source level
  - Virtual node descriptor specified
  - Constrained deployment with P2P deployment
- Mechanism for specifying environmental requirements:
  - **Defined by designer**
    - Specify minimun application deployment requirements
  - **Fulfilled by deployers**
    - Apply optimal configuration that fulfills requirement
- Work for component-based and object-based applications

P2P + Fault Tolerance + Load Balancing



With P2P: 5 clusters + INRIA lab desktops = 1007 CPUs