Declarative Security for GRID Applications: ProActive

Denis Caromel, Arnaud Contes

www.inria.fr/oasis/ProActive

OASIS Team

INRIA -- CNRS - I3S -- Univ. of Nice Sophia-Antipolis

1. Introduction to the GRID
2. ProActive: Remote Objects, Groups, Mobile Objects, Graphical Interface (IC2D), XML Deployment,
3. Declarative Security
4. Demonstration

1. Grid and the Internet

GRID definition:

- GRID = electric network in the US

A gripping idea:

Like electricity, computer cycles cannot be stored, if not used they are lost

A definition:

Grid is a parallel and distributed system that enables the use, sharing, selection, and aggregation of resources across multiple administrative domains based on their availability and capability.

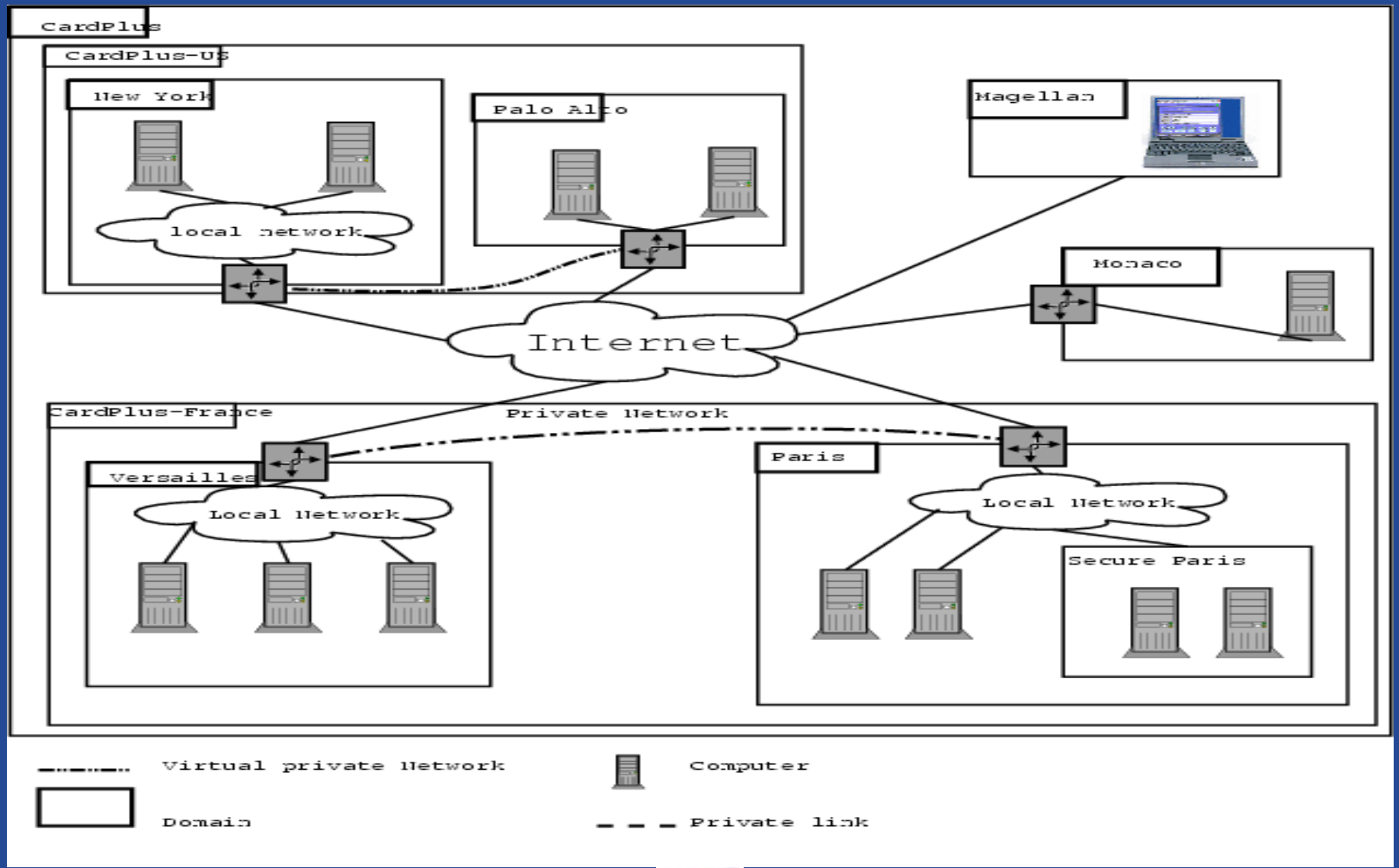
Not limited to cycles:

SECURITY ISSUES

Computational GRID, Data GRID

Inter, Intra-company, but multi-locations Grid (computational, and data)

Hierarchical Domains for Internet Grid



Issues at hand for Grid Security

Authentication of Computers, Users, and Applications

Authentication, Integrity and Confidentiality (AIC) of communications

Creation, connection to, and monitoring of activities

Hierarchical domains

Security Policies: Application, Domain, (sub-domain), … High-level!

Variation in Grid network links :

LAN, Wireless (Wifi, GPRS/UMTS), VPN, Internet, or … unknown !

Variation in deployment, but maintain as much as possible performance

2. 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**
- **Asynchronous Communications with synchro: automatic Futures**
- **Group Communications, Migration (mobile computations)**
- **XML Deployment Descriptors**
- **Interfaced with various protocols:** `rsh,ssh,LSF,Globus,Jini,RMIregistry`
- **Visualization and monitoring: IC2D**

In the www. ObjectWeb .org Consortium (Open Source middleware)
since April 2002 (**LGPL license**)

ProActive : Creating active objects

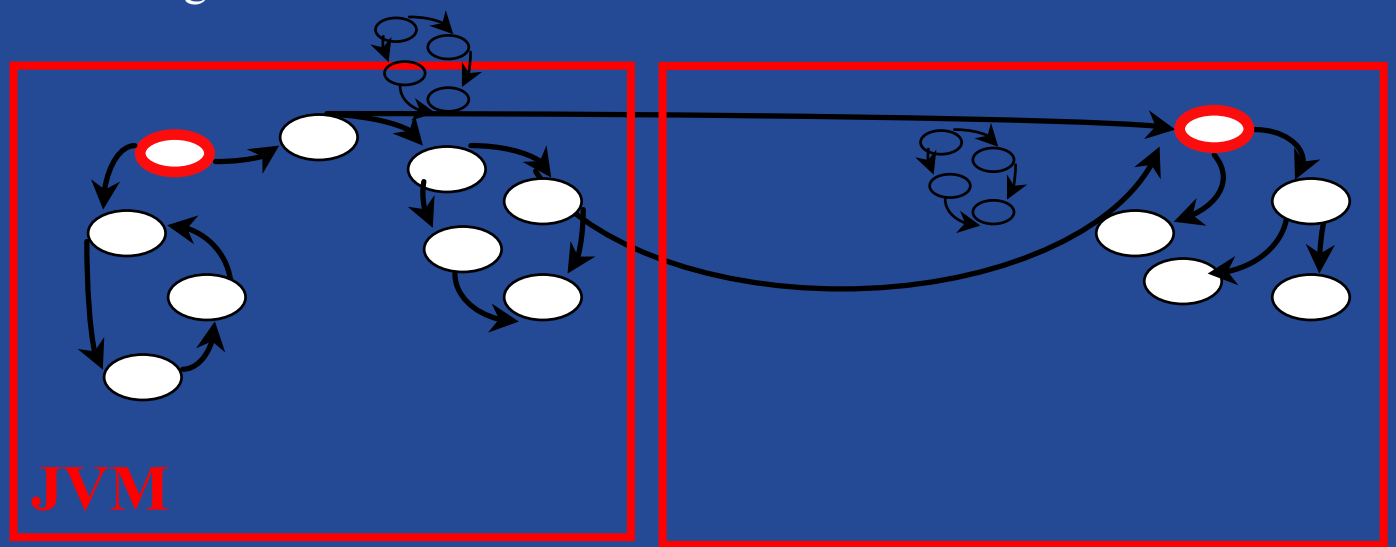
An object created with `A a = new A (obj, 7);`
can be turned into an active and remote object:

- **Instantiation-based:**

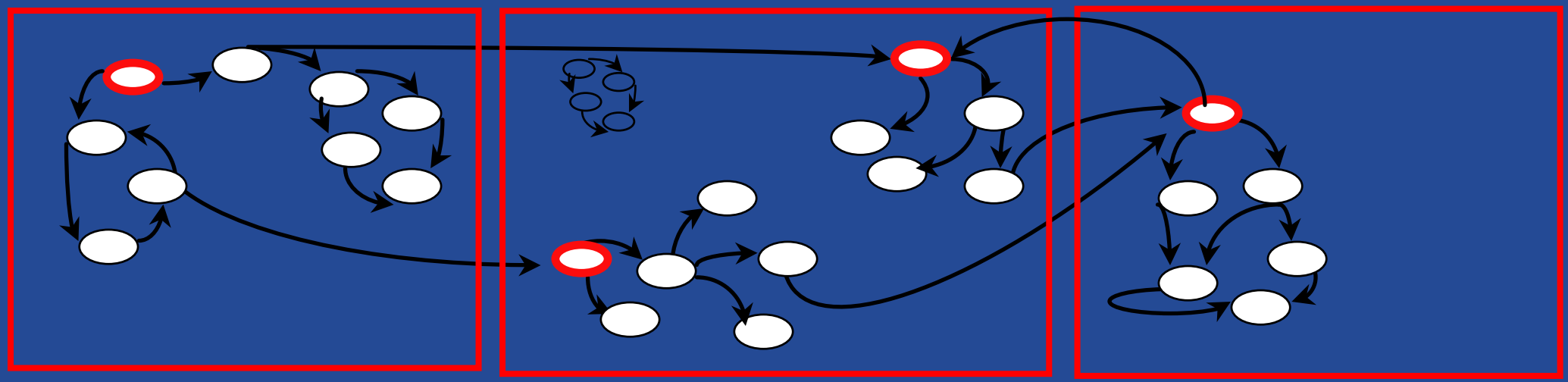
```
A a = (A)ProActive.newActive(«A», params, node);
```

The most general case.

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



Standard system at Runtime

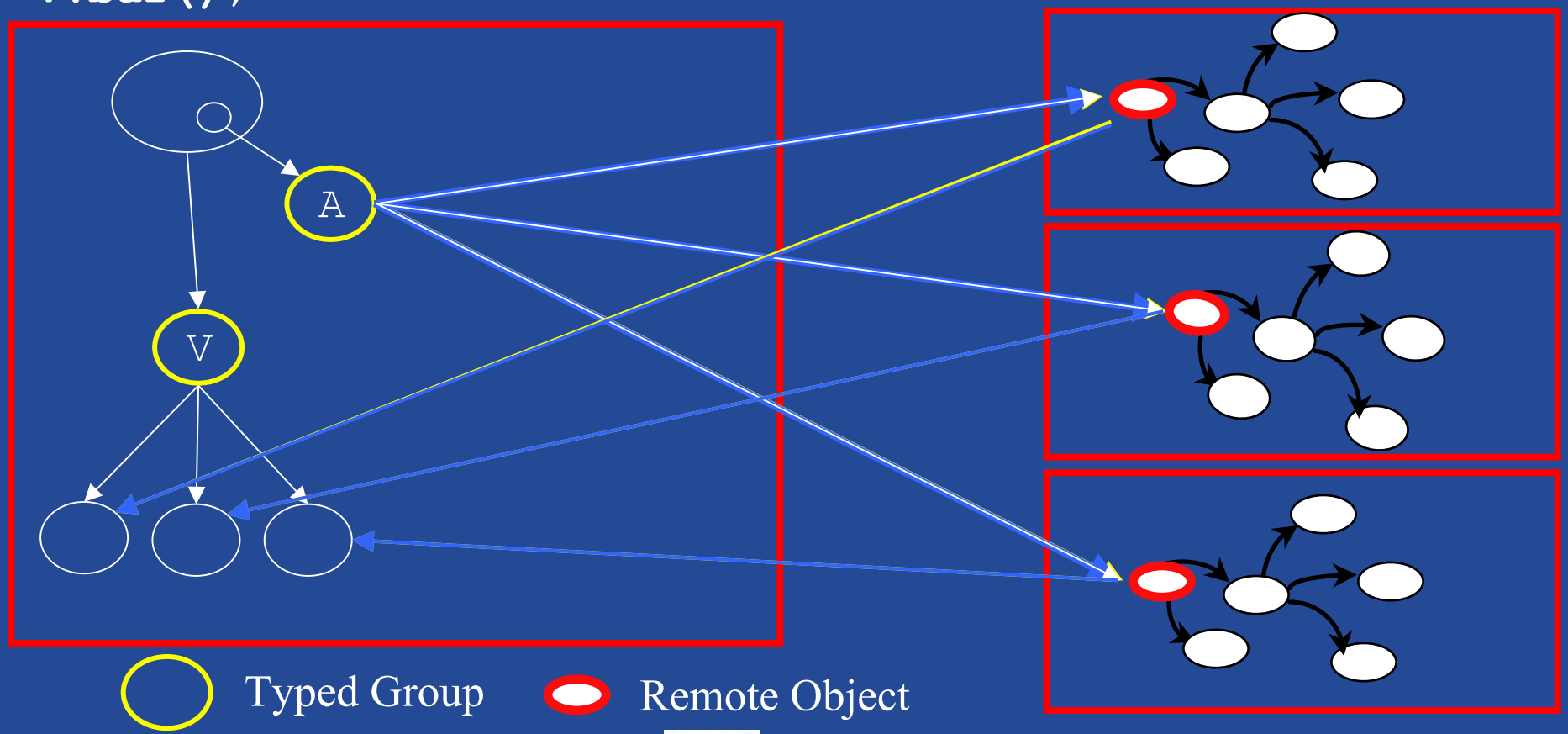


No sharing between activities

ProActive: Groups

Typed and polymorphic Groups of active and remote objects

```
A ag = newActiveGroup («A»,…,Nodes)
V v = ag.foo(param);
v.bar();
```



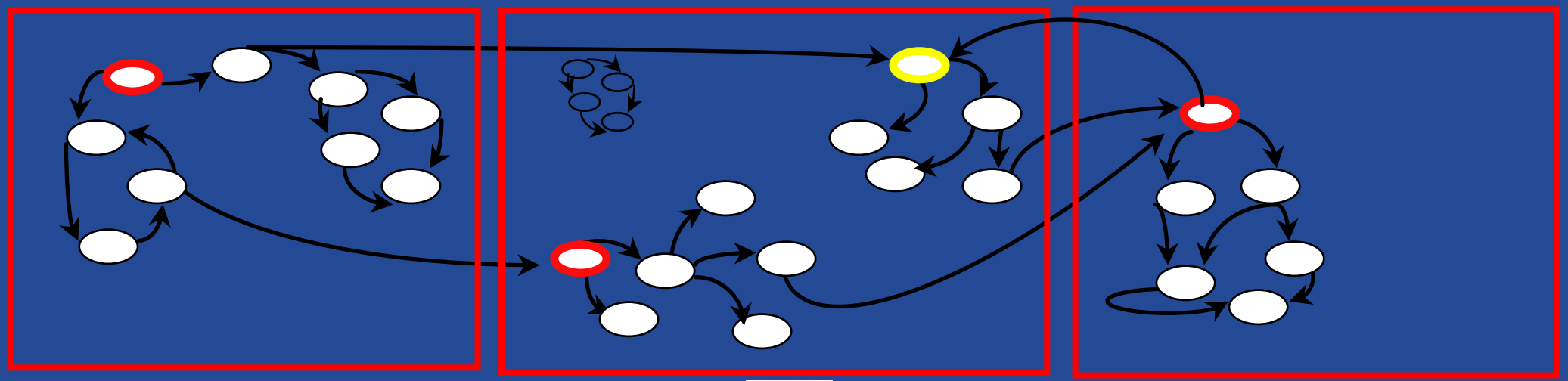
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



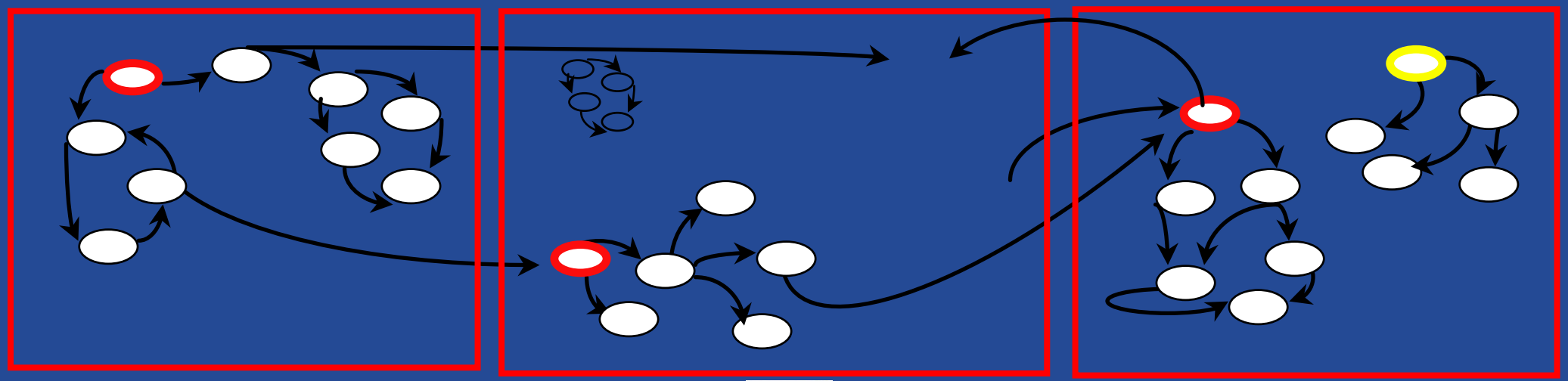
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



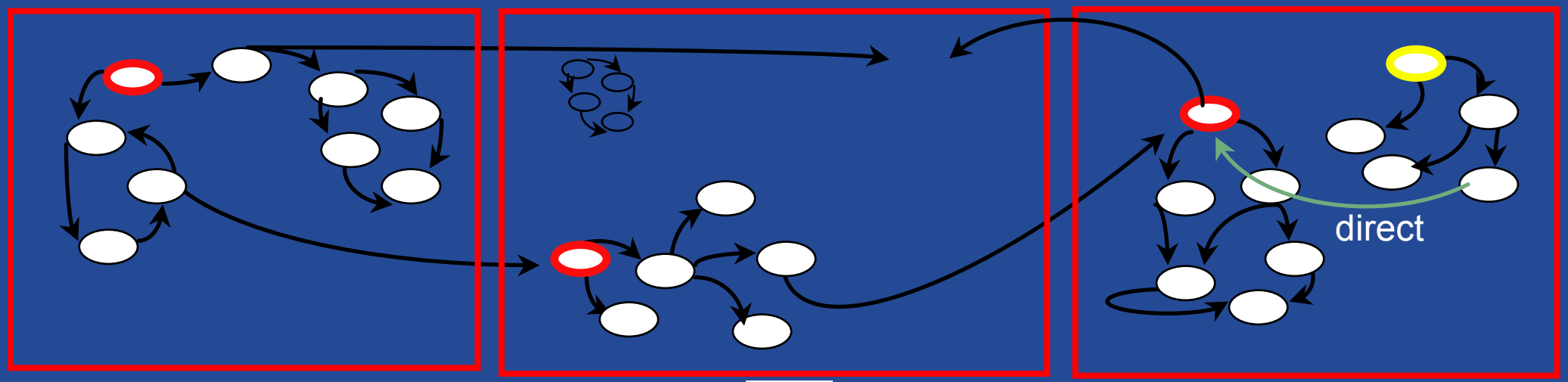
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



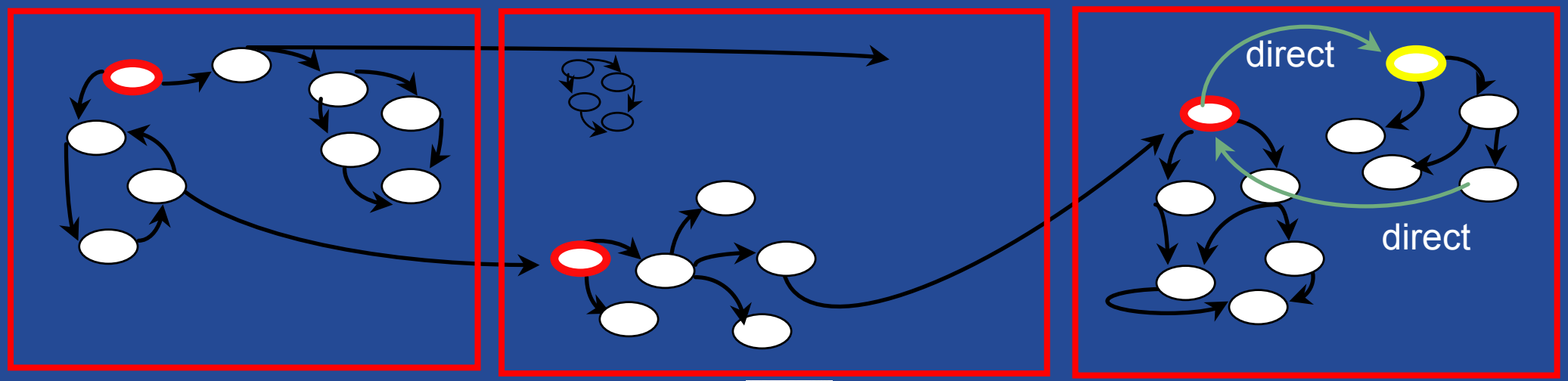
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



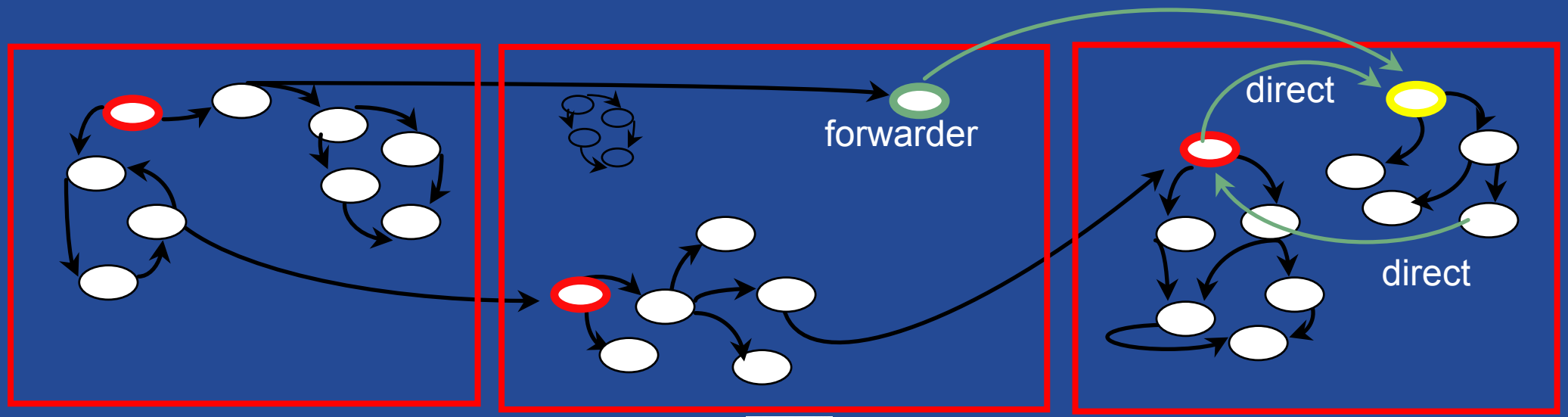
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



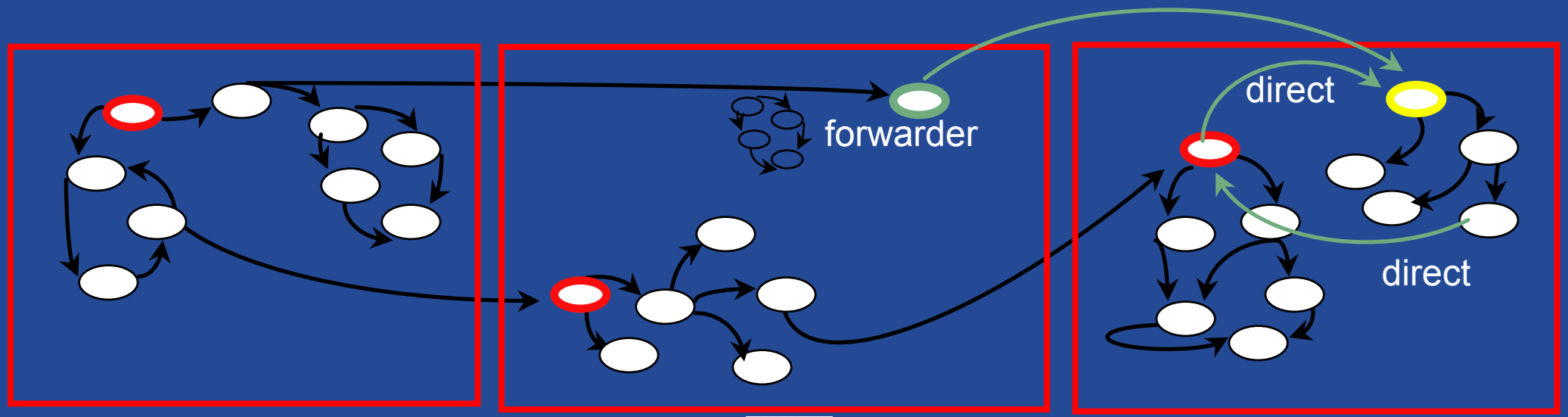
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



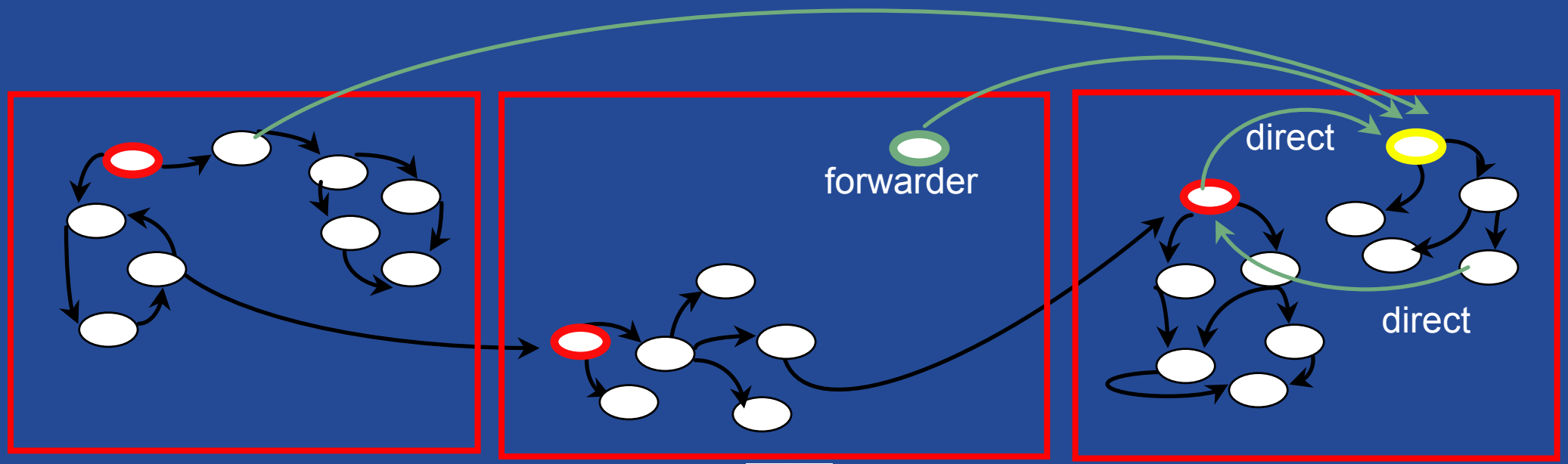
ProActive : Migration of active objects

Migration is initiated by a primitive: **migrateTo**

The active object migrates with: pending requests, objects, futures

Automatic and transparent forwarding of: requests, replies

Remote references remain valid



ProActive: Abstract Deployment Model

A key principle:

- Abstract Away from source code:
  - Machine names, Creation Protocols, Lookup and Registry Protocols

In program source:

- Virtual Node (VN, a string name):

In XML descriptors:

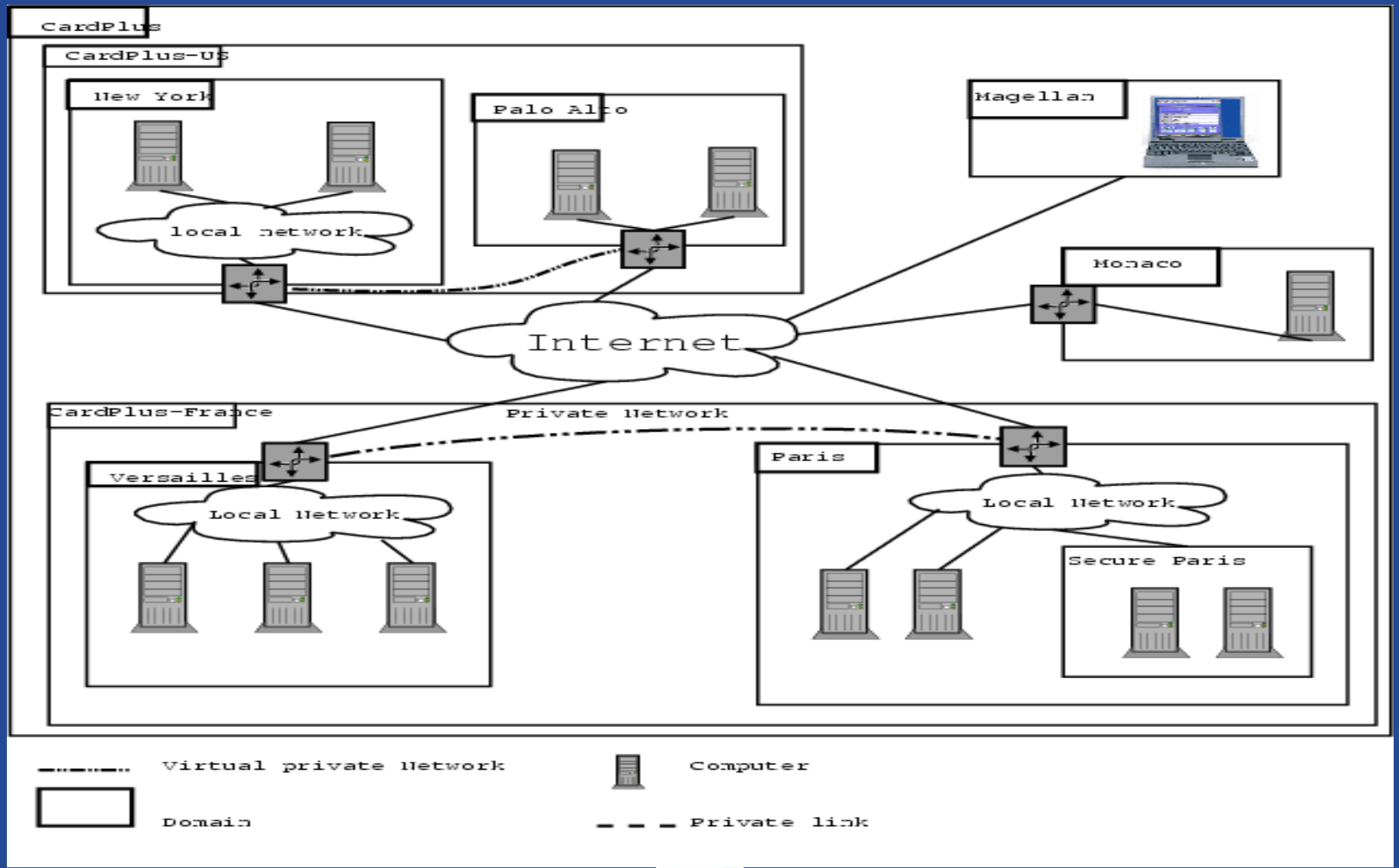
- Mapping of VN to JVMs (leads to Node in a JVM on Host)
- Register or Lookup VNs
- Create or Acquire JVMs

Program Source |----------| Descriptor (RunTime) |----------|

Activities (AO) --> VN VN --> JVMs --> Hosts

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

Hierarchical Domains for Internet Grid



Descriptors: Mapping Virtual Nodes

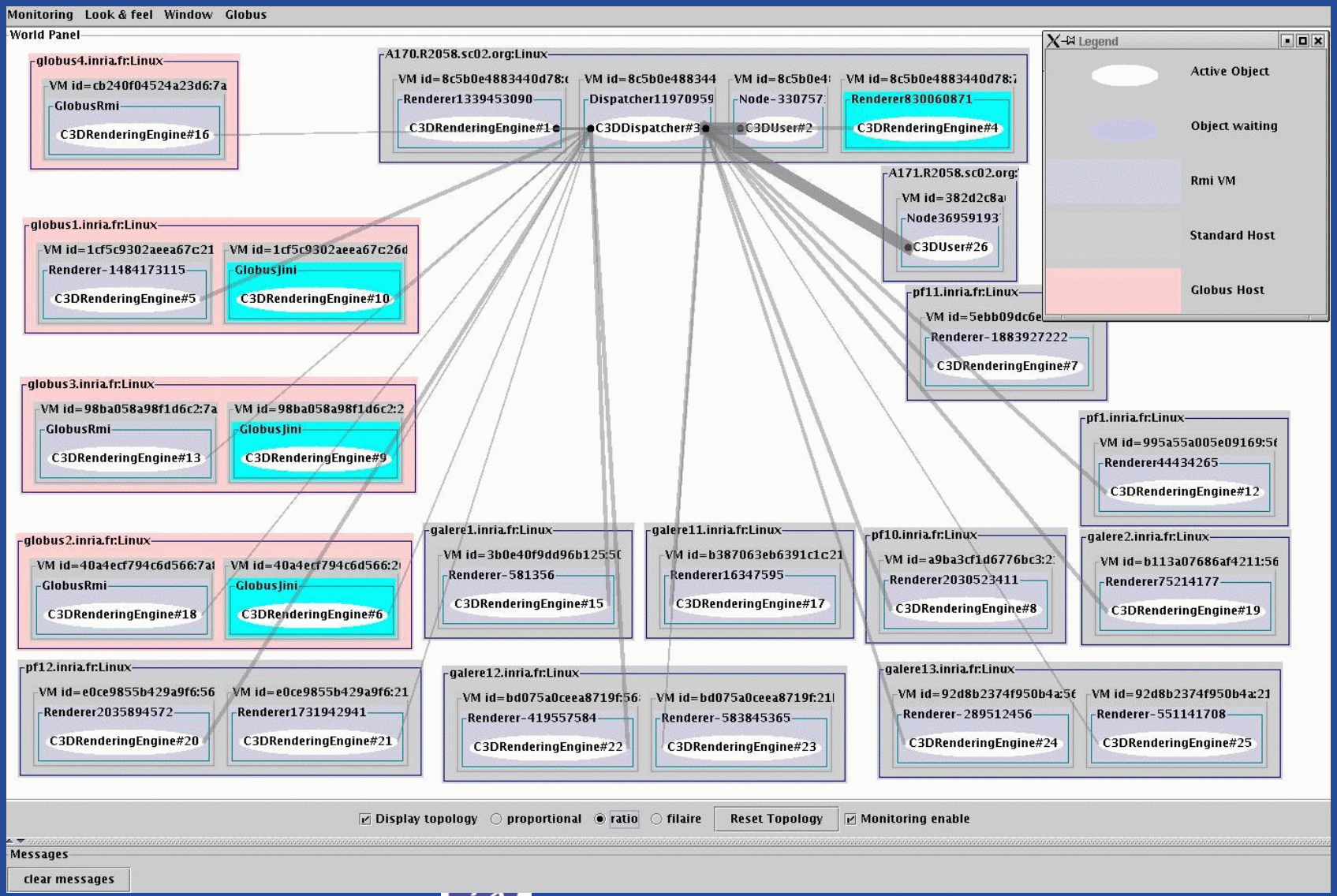
Example of an XML file descriptor:

```
Component Dependencies:
   Provides: …  Uses: ...
VirtualNodes:
   Dispatcher  <RegisterIn RMIregistry, Globus, Grid Service, … >
   RendererSet
Mapping:
   Dispatcher --> DispatcherJVM
   RendererSet --> JVMset
JVMs:
   DispatcherJVM = Current  // (the current JVM)
   JVMset=//ClusterSophia.inria.fr/ <Protocol GlobusGram … 10 >
...
```

Monitoring of RMI, Globus, Jini, LSF cluster Nice -- Baltimore

IC2D

Width of links proportional to the number of com-munications



3. Declarative Security

What's a secured ProActive application?

- Composed of 'classic' active objects, no change in sources
- Using
  - Public Key Infrastructure, X.509 Identity Certificates, Access control lists
  - XML description language
- PKI 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

- Mobility compliant

Security Rule

```
Entity -> Entity : Interactions # Security Attributes
```

Entities :

- Domain
- User
- Virtual Node
- Active Object

Each entity owns a certificate and depends on a Certification Authority.

Interactions :

- JVMCreation
- NodeCreation
- CodeLoading
- ActiveObjectCreation
- ActiveObjectMigration
- Request (Q)
- Reply (P)
- Listing

Attributes :

- Authentication (A)
- Integrity (I)
- Confidentiality (C)

Each attribute can be :

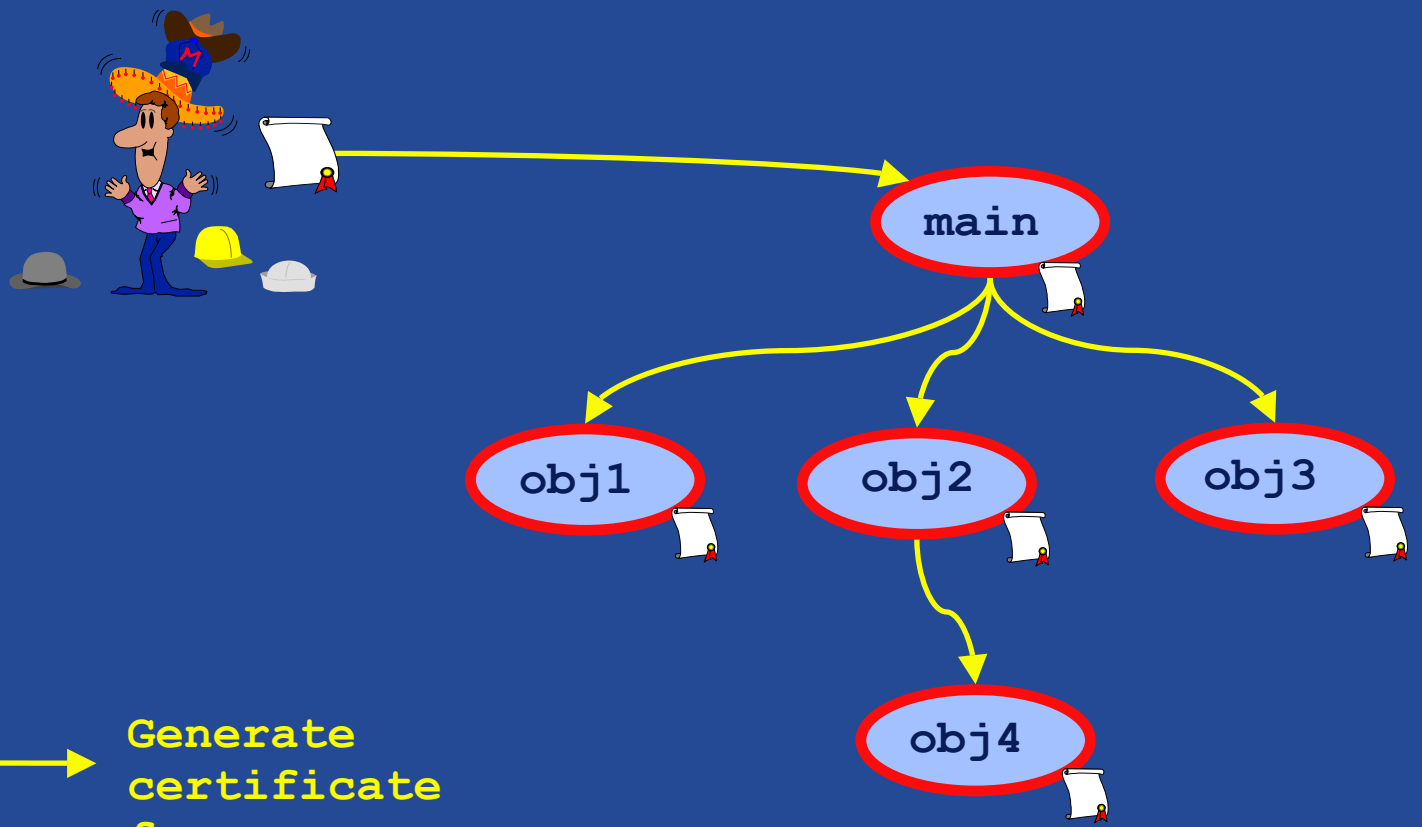
- Required (+)
- Optional (?)
- Disallowed (-)

Descriptors: Mapping Virtual Nodes

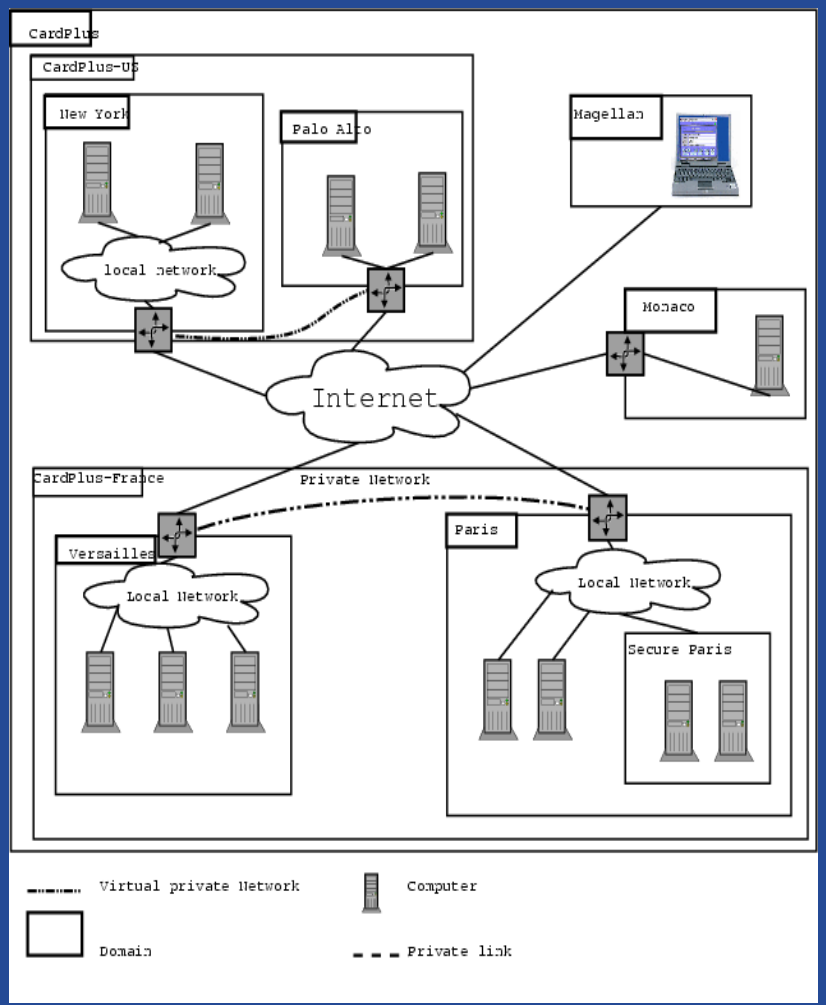
Example of an XML file descriptor:

```
VirtualNodes:
  Dispatcher  <RegisterIn RMIregistry, Globus, Grid Service, … >
  RendererSet
SECURITY:
   VN [Renderer]          -> VN [Dispatcher] : Q,P  #  [?A,?I,?C]
   VN [Dispatcher]        -> VN [Renderer]   : Q,P  #  [?A,?I,?C]
   Domain [CardPlus]      -> VN [Dispatcher] : Q,P  #  [+A,?I,?C]
Mapping:
  Dispatcher --> DispatcherJVM
  RendererSet --> JVMset
JVMs:
  DispatcherJVM = Current  // (the current JVM)
  JVMset=//ClusterSophia.inria.fr/ <Protocol GlobusGram … 10 >
```

Certification Chain

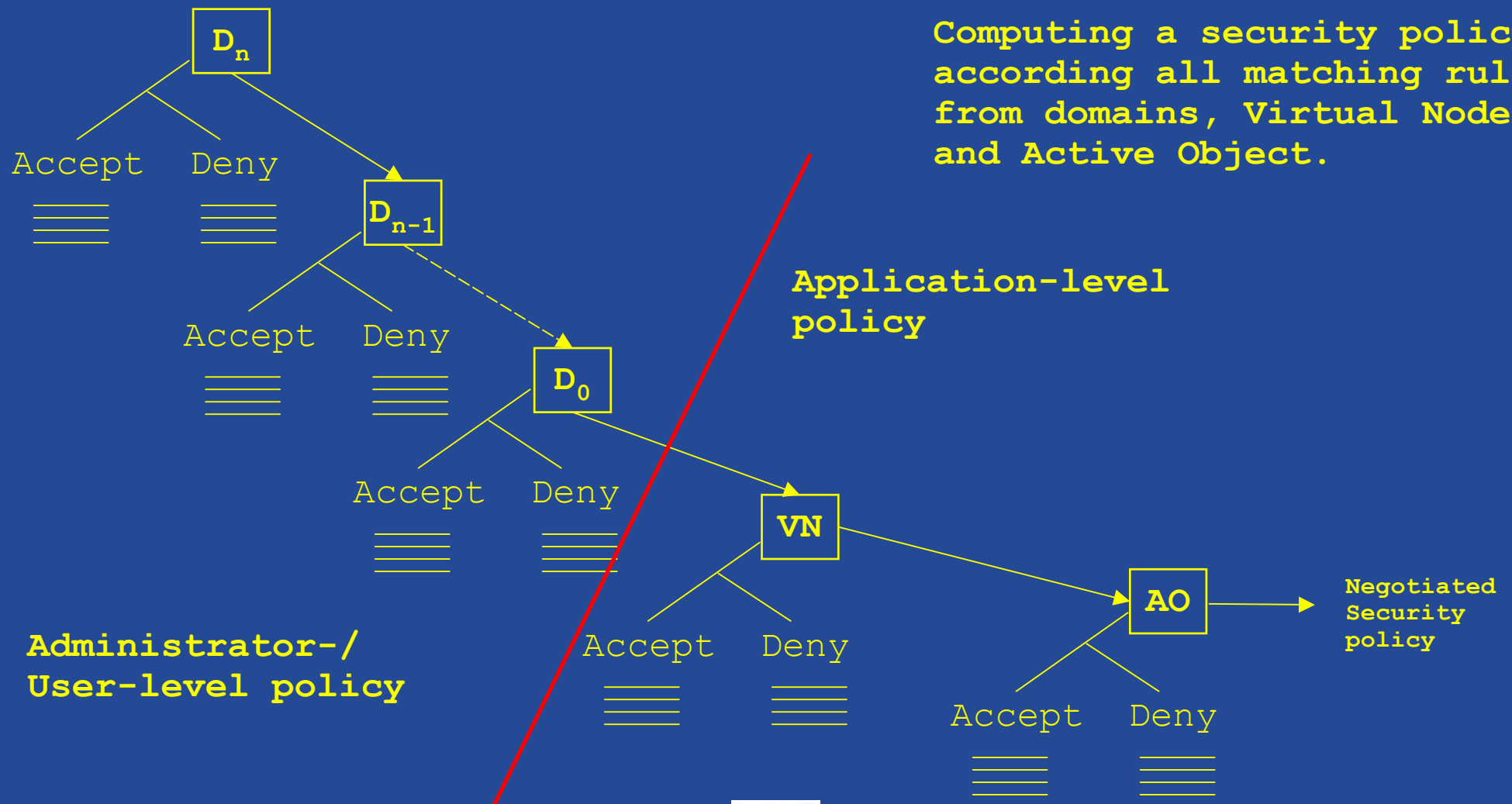


Hierarchical Security Domains



- Logical way to group many entities that have the same security needs.
- Domains are hierarchical.
- Sub-domains inherits parent's security policies.
- Default : Sub-domains cannot weaken parent's security policies.
- 'Can override' : a domain authorizes an entity to override its policies (doPrivileged)

Multi-level Policies



Computing a security policy according all matching rules from domains, Virtual Node and Active Object.

Combining Policies

- Search for the most specific rule in each domain (if exists).
- Retrieve all matching rules in the Domain hierarchy, the Virtual Node and the Active Object.
- Compute policies according to security attributes.

Sender \ Receiver	Required (+)	Optional (?)	Disallowed (-)
Required (+)	+	+	invalid
Optional (?)	+	?	-
Disallowed (-)	invalid	-	-

Migration & Security

- Migration can invalidate negotiated policies :
  - migration to a node of the same domain
  - migration to a node of another domain

===> New Security Negotiation

4. Demonstration:
Declarative Security with Mobility

C3D : Collaborative 3D renderer in //

a standard ProActive application

with the IC2D monitor

IC2D: Interactive Control & Debug for Distribution

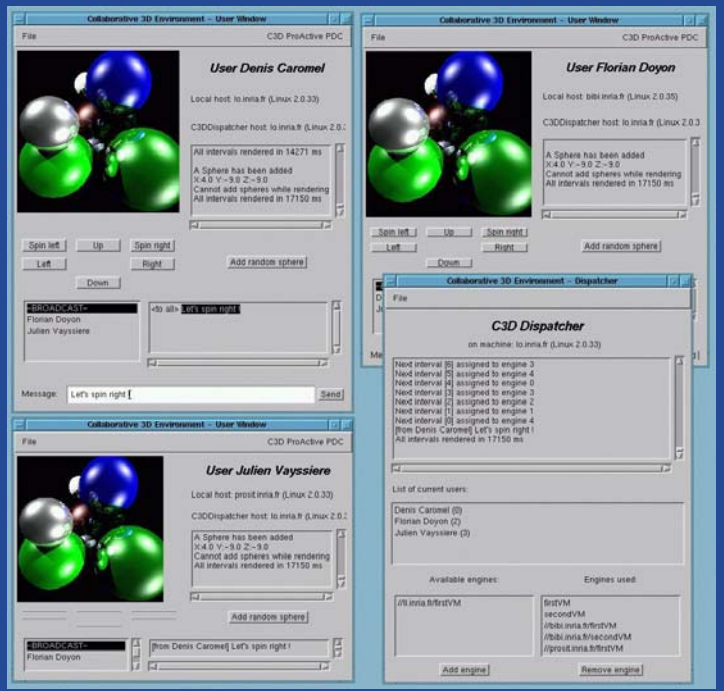
work with any ProActive application

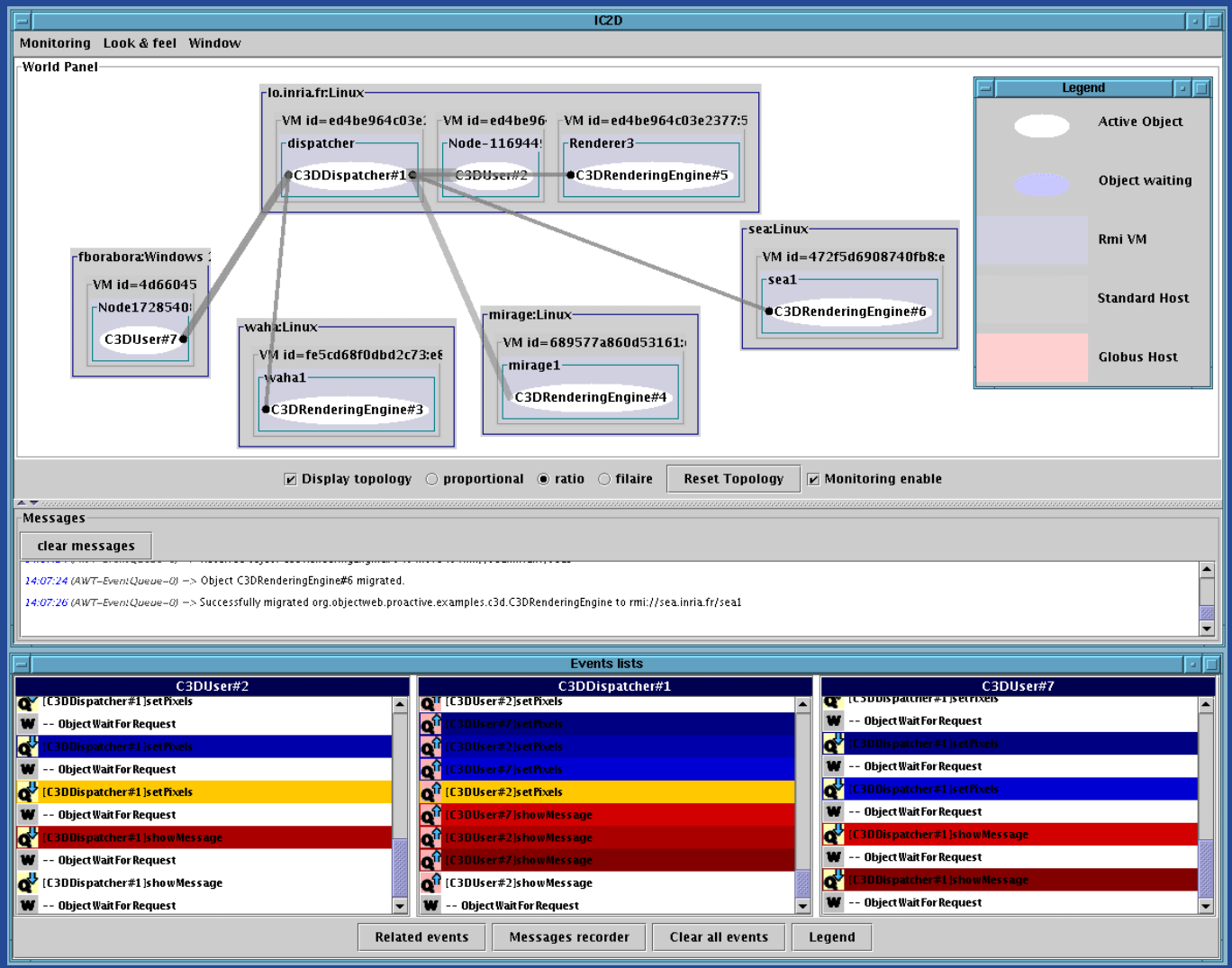
Features:

Graphical and Textual visualization

Monitoring and Control

C3D : Collaborative 3D renderer in //





Comparisons with Related Work

- ProActive Basic Features
  - Authentication of users and applications
  - Authentication, integrity and confidentiality of communications
  - Security model for fully mobile applications
  - Dynamically negotiated policies, non-functional security
  - Logical representation : security is easily adaptable to the deployment
- Security Frameworks
  - .Net, Legion, Globus: no notion of application mobility
  - Globus: Grid Security Infrastructure (GSI):
    single sign on, delegation, and credential mapping,
    but no high-level control, no easy encryption of communications
- Security in Agent platforms
  - Ajanta, Mole, Aglets, MAP: limited code mobility (fixe host + mobile agent)

Conclusion

ProActive Perspectives :

- Group communication (key management, find common policy)
- Sandboxing of nodes
- Role-based access control
- Components (Distributed, Parallel, Hierarchical) and Security

General Perspectives:

- OGSA Security: Open Grid Services Architecture
  - Globus new open architecture, Web Services based
- Security code no longer instantiated within the middleware:

  the middleware (and applications) calls external Web Security Services

  ⇒ but high-level abstractions, still needed (domain, application-level)

  ⇒