

Being



About Adaptive Grid Middleware

Denis Caromel, et al.

www.inria.fr/oasis/ProActive

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

Sept. 30 2004

1. "Being Adaptive": Generalities
2. **ProActive**: Adaptive Features
3. Adaptive Genes: Active Objects-Components vs. MPI



A Few Generalities on "Adaptive"



Dictionary Definitions

Adaptive, Adaptative:

Having a capacity for adaptation;

''The adaptive coloring of a chameleon''

==> An entity adapting to the environment

''Auto-adaptive'': ... No ! Pleonasm, already by definition

How to be adaptive?

Adaptable ==> Parameterized

First:

Good Parameterized Strategies and Protocols

– Design

– Model, Performance Evaluation, Simulation, Emulation, Benchmarks

Configurable ==> then can become: **Effectively Adaptive**



Dictionary Definitions (2)

ANTONYM: Maladaptive

Showing **faulty adaptation**

=> dysfunctional, nonadaptive -- of a trait or condition

Failing to serve an adjustive purpose;

==> Dysfunctional behavior

==> Poorly adjusted

So, if your Middleware is adaptive, it can indeed be
Maladaptive!

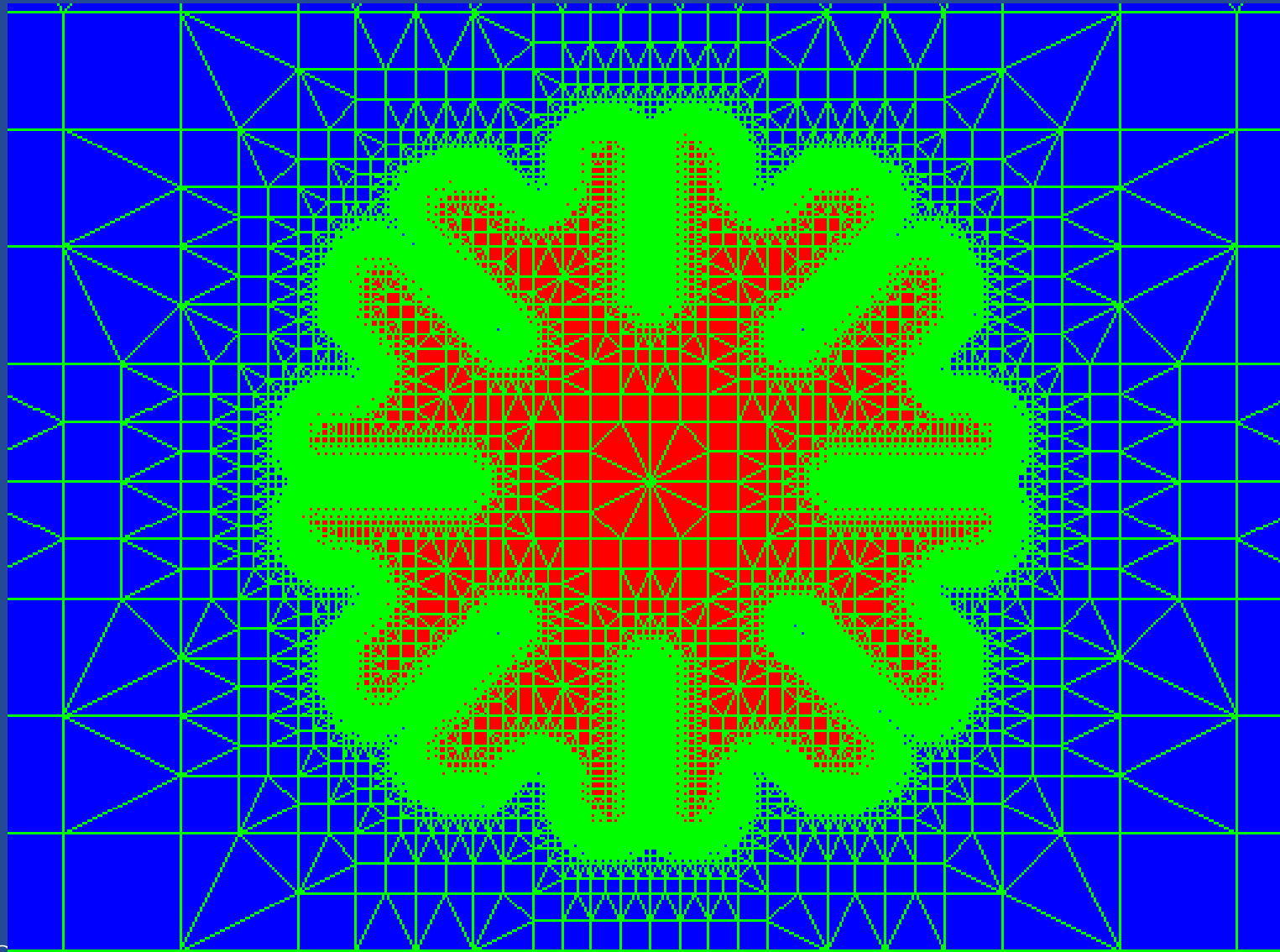


Adaptive Grids

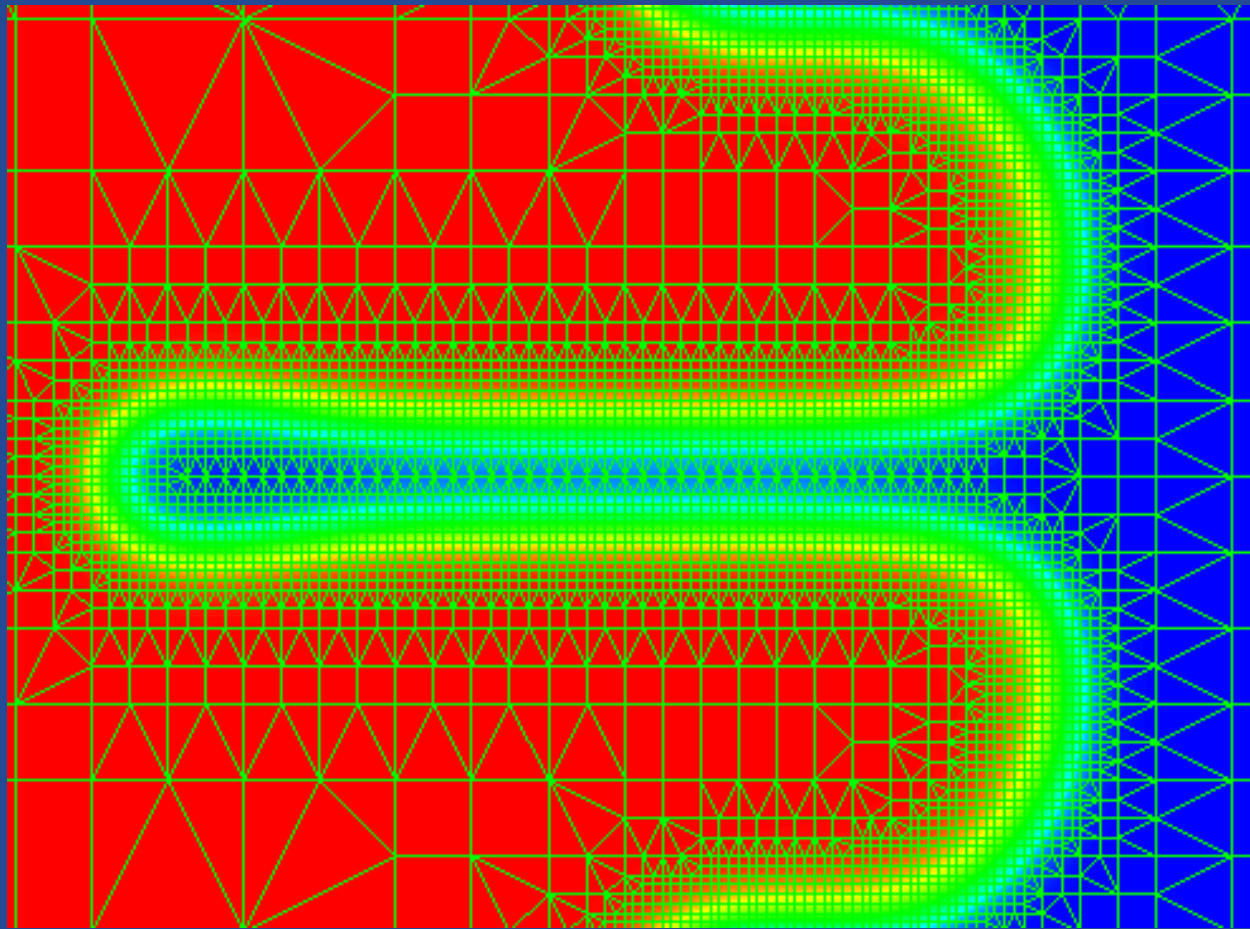
in pictures



What "Adaptive Grids" used to be



Zooming



Numerical Dendritic Growth (NDG):
Modeling Solidification using Phase-Field Equations
Solved by Adaptive Grid Methods

Adaptation in Grid applications:

Not only the middleware is being Adaptive

Many applications require adaptive strategies, e.g.:

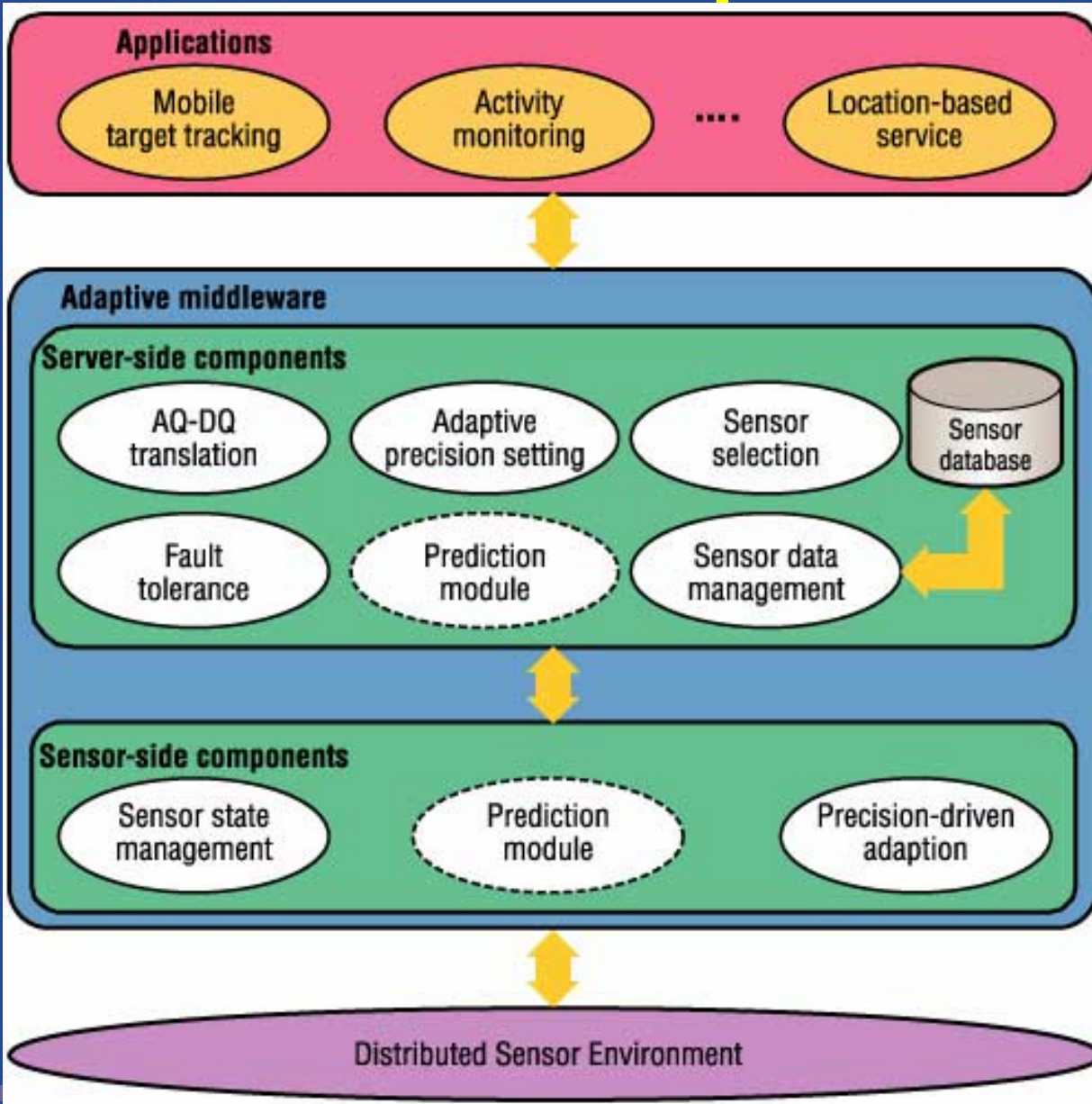
- Adaptive Numerical Method ==> Mesh Refinement
- Adaptive Multi-scale Bio-simulation
- Adaptive Discretisation Schemes
- ...

How to make sure

**Adaptive Applications do not confuse Adaptive Middelwares ?
ending up into ... Maladaptive Grid !**



What "Adaptive Grids" are now



Adaptive Middleware for
Distributed Sensor Environments

Xingbo Yu, Koushik Niyogi,
Sharad Mehrotra, Nalini
Venkatasubramanian

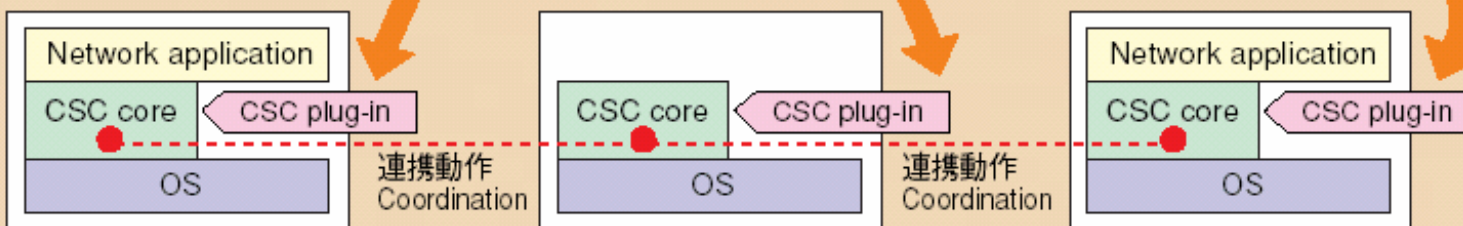
University of California, Irvine
IEEE Distributed System Online
May 2003

"Adaptive Grids" are complex systems

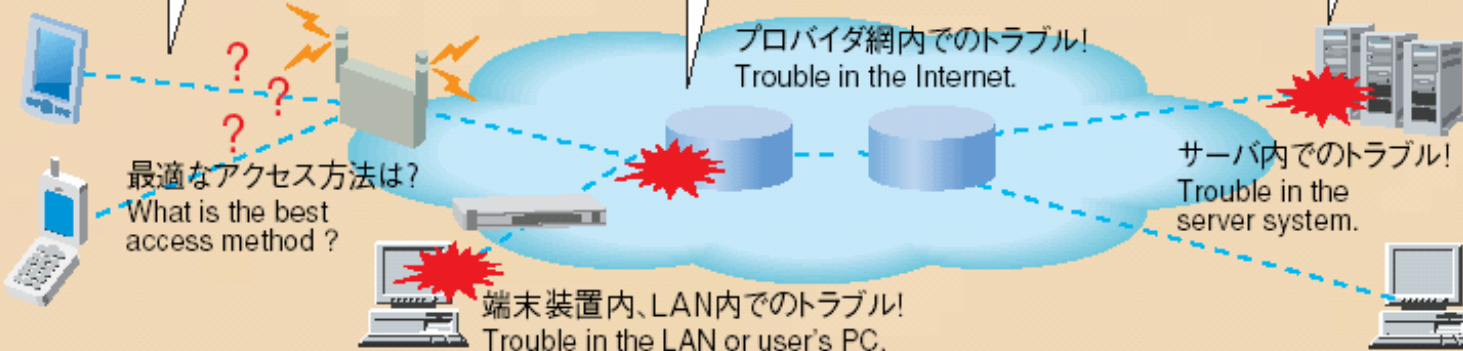
～通信のエンドエンドに介在するさまざまな要素を連携・協調させて快適な通信を実現するミドルウェア～
CSC : Communication Service Concierge



ネットワークを介したダウンロード、動的な制御機能の追加変更が可能
CSC plug-ins can be downloaded dynamically via the Internet.



CSC の連携動作によって、通信のエンドエンドでさまざまな問題を解決
CSC solves various problems in the end-to-end communication link with coordinating plug-in modules.



Adaptive
Communication
Control
Middleware
"CSC"

NTT
Cyber
Communication
Laboratory



Adaptation in Grid Middleware:

Adaptive strategies at many locations:

- Communication transports and strategies
- Discovery, Localization, Routing
- Fault-Tolerance
- Managing Disconnections
- Security
- Buffering
- Scheduling, Load Balancing
- ...

How to make sure

**Adaptive Strategy N do not confuse Adaptive Strategy M ?
ending up into ... Maladaptive Grid !**



2. Adaptivity in

ProActive

Programming

Wrapping

Composing

Deploying



ProActive:

A Java API + Tools for Parallel, Distributed Computing

A uniform framework: **An Active Object pattern**

A formal model behind: **Determinism (POPL'04)**

Programming Model:

- Remote Objects (**Classes, not only Interfaces, Dynamic**)
- Asynchronous Communications, Wait-By-Necessity
- Groups, Mobility, Components, Security

Environment:

- XML Deployment Descriptors
- Interfaced with: **rsh, ssh, LSF, PBS, Globus, Jini, SUN Grid Engine**
- Graphical Visualization and monitoring: **IC2D**

In the www.ObjectWeb.org Consortium

(Open Source **LGPL**)



ProActive model

Java RMI (Remote Method Invocation = Object RPC = `o.foo(p)`)

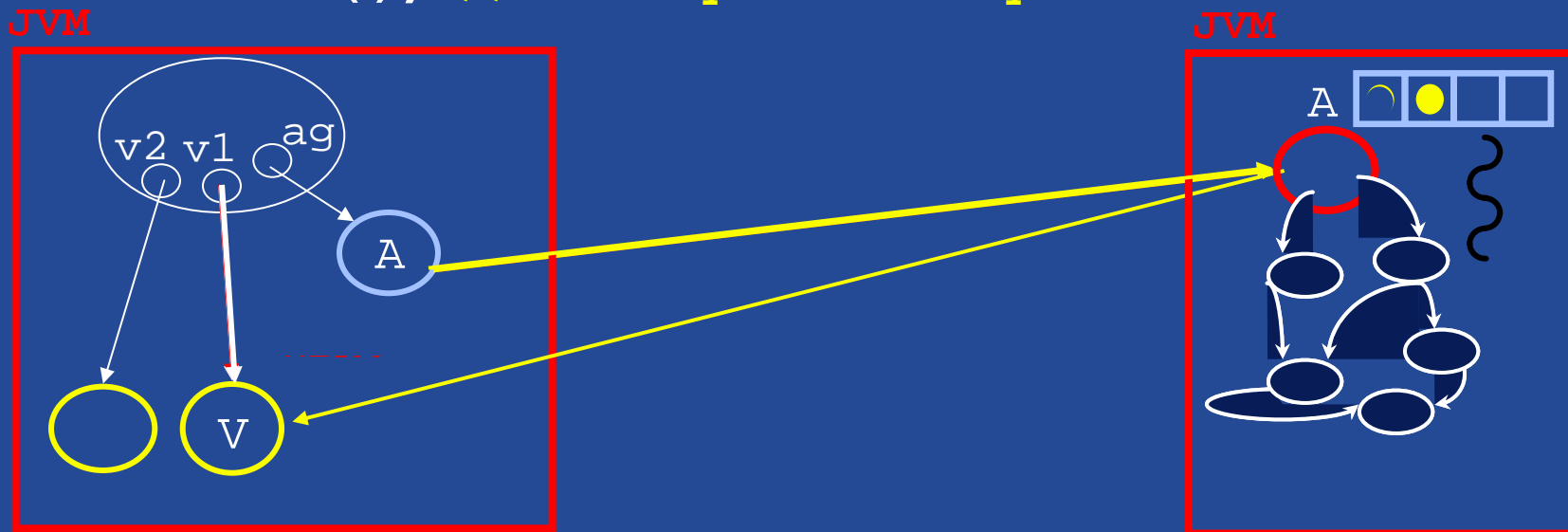
plus a few important features:

- Sequential Object: a single thread with default FIFO service
- No shared passive objects
- Asynchronous Method calls towards Active Objects:
 - Implicit Futures as method results
- Wait-By-Necessity:
 - Automatic wait upon a strict operation on an unknown future
 - First-Class Futures:
 - Futures can be passed to other activities
 - Sending a future to another machines is not blocking



ProActive : Active objects

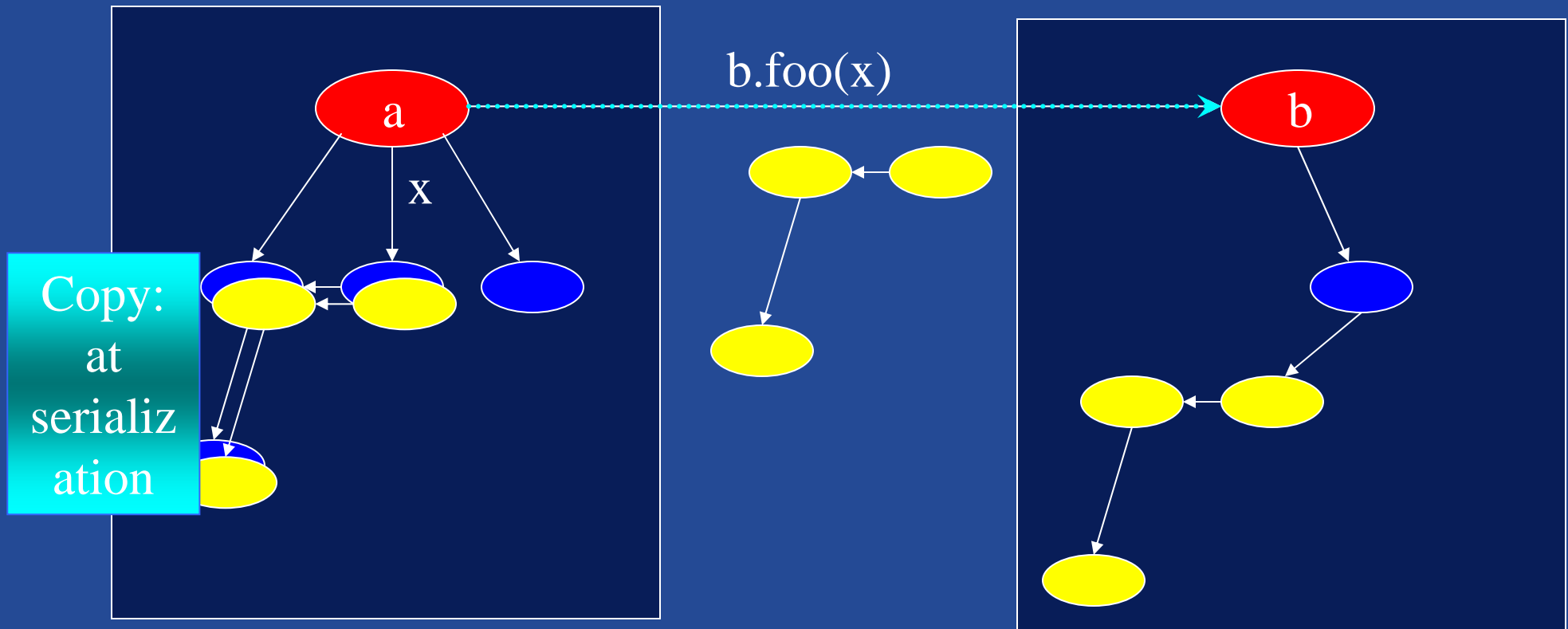
- A ag = `newActive` ("A", [...], VirtualNode)
- V v1 = ag.foo (param);
- V v2 = ag.bar (param);
- ...
- v1.bar(); //Wait-By-Necessity



Wait-By-Necessity
is a
Dataflow
Synchronization



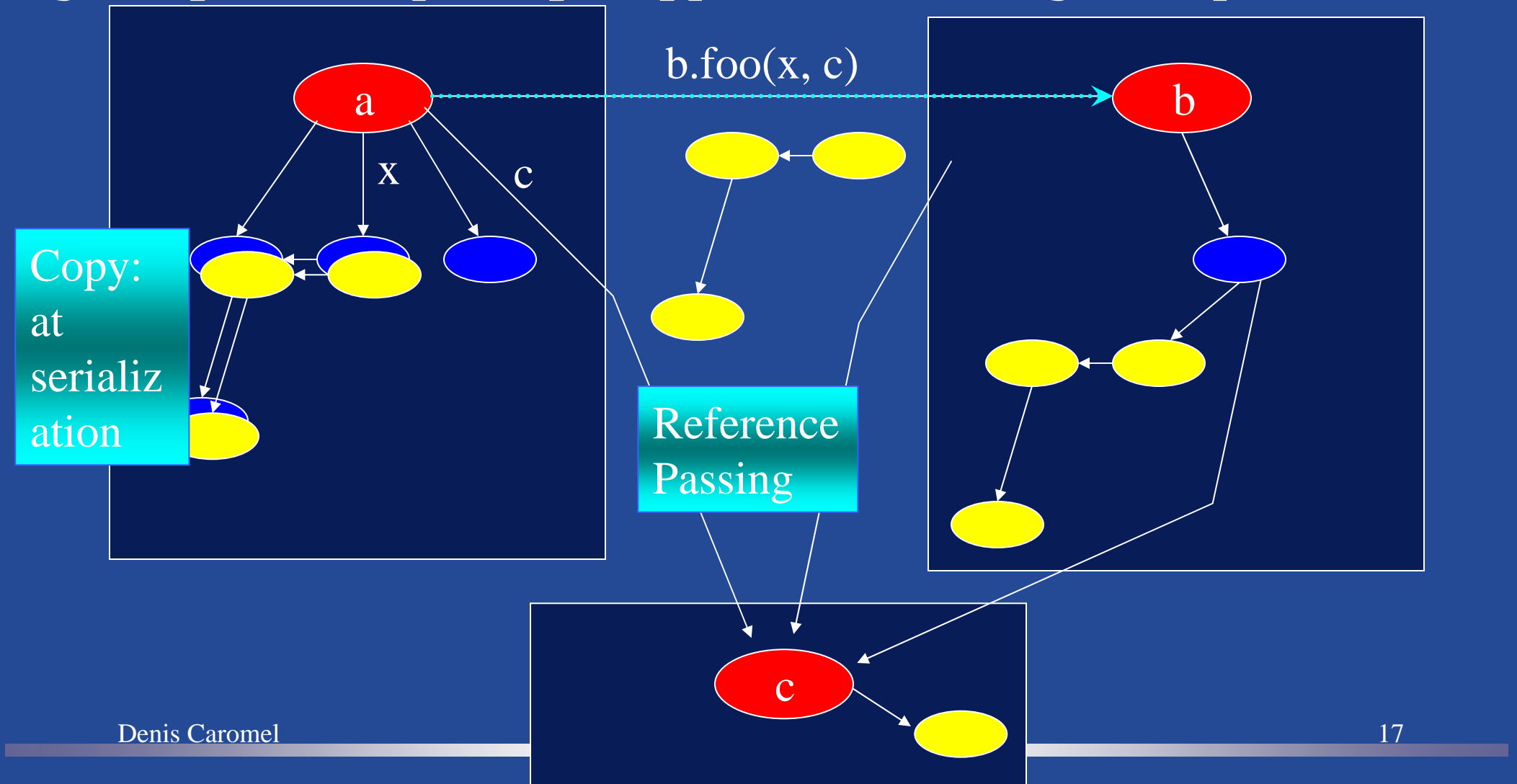
Call between Objects: Parameter passing: Copy of Java Objects



(Deep) Copies evolve independently -- No consistency

Call between Objects: Parameter Passing: Active Objects

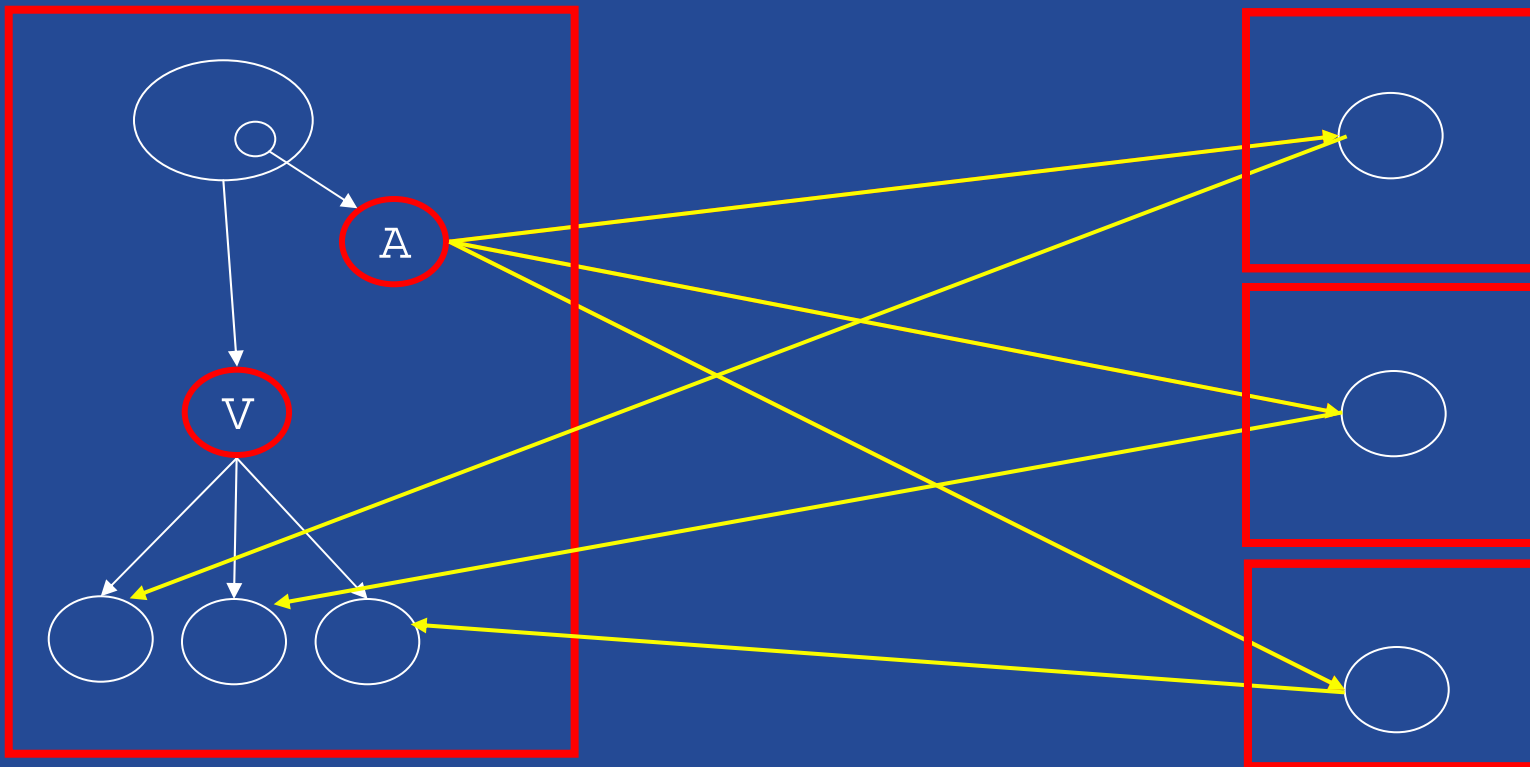
Object passed by Deep Copy - Active Object by Reference



Creating AO and Groups

```
A ag = newActiveGroup ("A", [...], VirtualNode)  
V v = ag.foo(param);  
...  
v.bar(); //Wait-by-necessity
```

JVM



Object-Oriented
Typed Group Communications

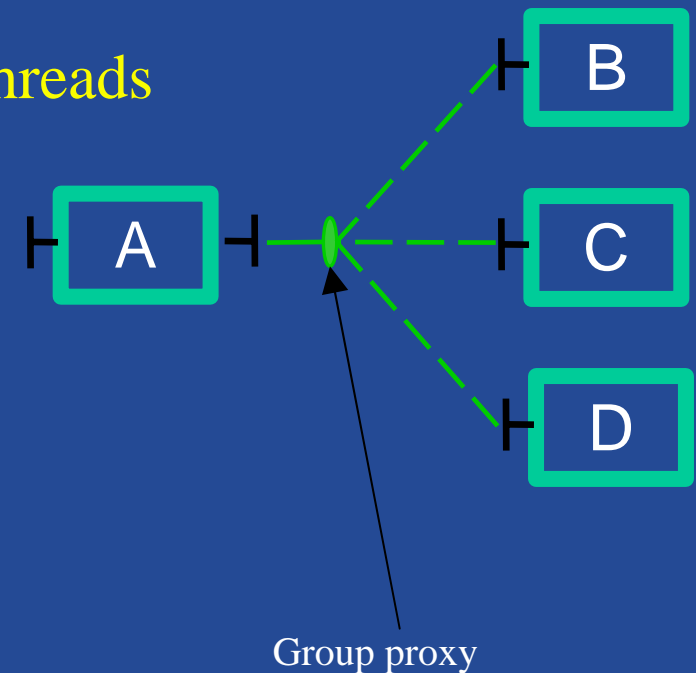
Group, Type, and Asynchrony
are crucial for Cpt. and GRID



Adaptive Feature 1: Parallel Group Communications

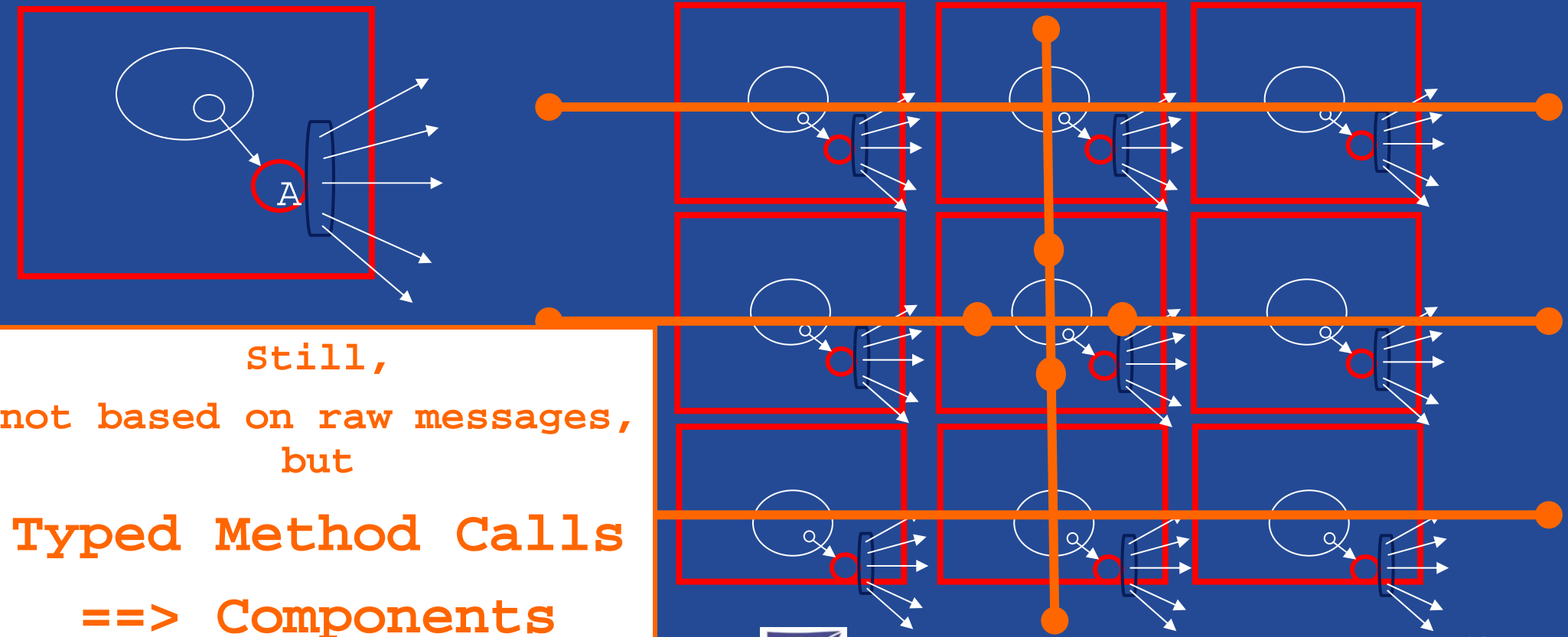
Adaptive strategy to manage the number of threads

Execution of N calls in //



OO SPMD

- A ag = `newSPMDGroup ("A", [...], VirtualNode)`
 - // In each member
 - `myGroup.barrier ("2D"); // Global Barrier`
 - `myGroup.barrier ("vertical"); // Any Barrier`
 - `myGroup.barrier ("north","south","east","west");`



IC2D: Interactive Control and Debugging of Distribution

The screenshot shows the IC2D monitoring application interface. At the top, there are menu items: "Monitoring", "Look & feel", "Window", and "Globus". Below the menu is the "World Panel" which displays a hierarchical tree of objects. The tree is organized into three main VMs: "mirage:Linux", "pagode:Linux", and "owenii:Linux".

- mirage:Linux** (VM id=689577a860d53161:ded) contains "Containers-175091467" which includes ActivePrimeContainer#31, #32, #33, and #34.
- pagode:Linux** (VM id=272cf6436bcff3fa:e859c) contains "Containers-728729810" which includes ActivePrimeContainer#39, #40, #29, and #41. A "NumberSource#27" object is also shown within this container group.
- owenii:Linux** (VM id=176643bd02a5bc92:ded) contains "Containers-88907713" which includes ActivePrimeContainer#35, #36, #37, and #38.

At the bottom of the window, there are control buttons: "Display topology" (checked), "proportional", "ratio", "filaire" (selected), "Reset Topology", and "Monitoring enable" (checked). A "Messages" panel at the bottom shows a log of events:

```
13:52:15 (AWT-EventQueue-0) => Received object ActivePrimeContainer#29 to move to rmi://pagode/Containers-728729810
13:52:15 (AWT-EventQueue-0) => Object ActivePrimeContainer#29 migrated.
13:52:16 (AWT-EventQueue-0) => Successfully migrated
org.objectweb.proactive.examples.eratosthenes.ActivePrimeContainer to rmi://pagode/Containers-728729810
```

The "World Panel Legend" window provides a key for the visual elements used in the main monitoring interface. It is divided into three sections:

- Active objects:** Shows five states with corresponding colored ovals: "Active by itself" (green), "Serving request" (white), "Waiting for request" (grey), "Waiting for result (wait by necessity)" (yellow), and "Migrating" (red).
- Pending Requests:** Shows a white oval with green and red dots. A legend below indicates the number of pending requests: 1 (green square), 5 (red square), and 50 (blue square).
- Nodes:** Shows two types: "RMI Node" (light blue rectangle) and "Jini Node" (cyan rectangle).
- Hosts:** Shows two types: "Standard Host" (light pink rectangle) and "Globus Host" (pink rectangle).

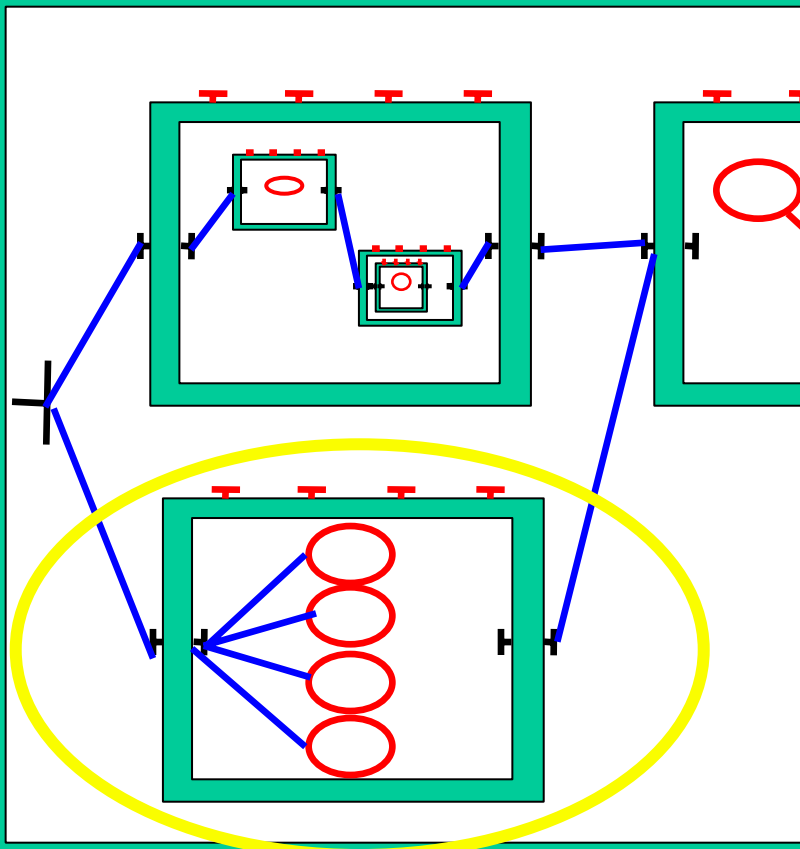
With any ProActive application
Features:
Graphical and Textual visualization
Monitoring and Control



Distributed Components

Graphical Composition, Monitoring, Migration

Composition View



Monitoring Look & feel Window Globus

World Panel

The monitoring window displays a network of distributed components. Each component is represented by a box containing its name, VM ID, and specific roles. Lines connect these components, showing their interdependencies. A green box highlights a row of components at the bottom, and several of these are circled in red.

Component Name	VM ID	Roles
-globus4.inria.fr:Linux-	VM id=cb240f04524a23d6:7a	-GlobusRmi- C3DRenderingEngine#16
A170.R2058.sc02.org:Linux-	VM id=8c5b0e4883440d78:c	-Renderer1339453090 C3DRenderingEngine#1
	VM id=8c5b0e488344	Dispatcher11970959
	VM id=8c5b0e4	-Node-330757- C3DUser#2
	VM id=8c5b0e4883440d78:7	Renderer830060871 C3DRenderingEngine#4
A171.R2058.sc02.org:	VM id=382d2c8a	Node36959193 C3DUser#26
pf11.inria.fr:Linux-	VM id=5ebb09dc6e	Renderer-1883927222 C3DRenderingEngine#7
-globus1.inria.fr:Linux-	VM id=1cf5c9302aeea67c21	Renderer-1484173115 C3DRenderingEngine#5
	VM id=1cf5c9302aeea67c2d	GlobusJmi C3DRenderingEngine#10
-globus3.inria.fr:Linux-	VM id=98ba058a98f1d6c:27a	GlobusRmi C3DRenderingEngine#13
	VM id=98ba058a98f1d6c:22	GlobusJmi C3DRenderingEngine#9
-globus2.inria.fr:Linux-	VM id=40a4ecf794c6d566:7a	-GlobusRmi- C3DRenderingEngine#18
	VM id=40a4ecf794c6d566:2	GlobusJmi C3DRenderingEngine#6
-galere1.inria.fr:Linux-	VM id=3b0e40f9dd96b125:5c	Renderer-581356 C3DRenderingEngine#15
-galere11.inria.fr:Linux-	VM id=b387063eb6391c1c:21	Renderer16347595 C3DRenderingEngine#17
pf10.inria.fr:Linux-	VM id=a9ba3cf1d6776bc:3:2	Renderer2030523411 C3DRenderingEngine#8
pf12.inria.fr:Linux-	VM id=a0ce0855b420a066:56	Renderer2035804572 C3DRenderingEngine#2
	VM id=a0ce0855b420a066:21	Renderer1731942941 C3DRenderingEngine#21
-galere12.inria.fr:Linux-	VM id=8007300cc081125:8b	Renderer-419557584 C3DRenderingEngine#2
	VM id=8007300cc081125:21	Renderer-583845365 C3DRenderingEngine#23
-galere13.inria.fr:Linux-	VM id=824023274605811:57	Renderer-289512456 C3DRenderingEngine#2

Display topology proportional ratio filaire Monitoring enable

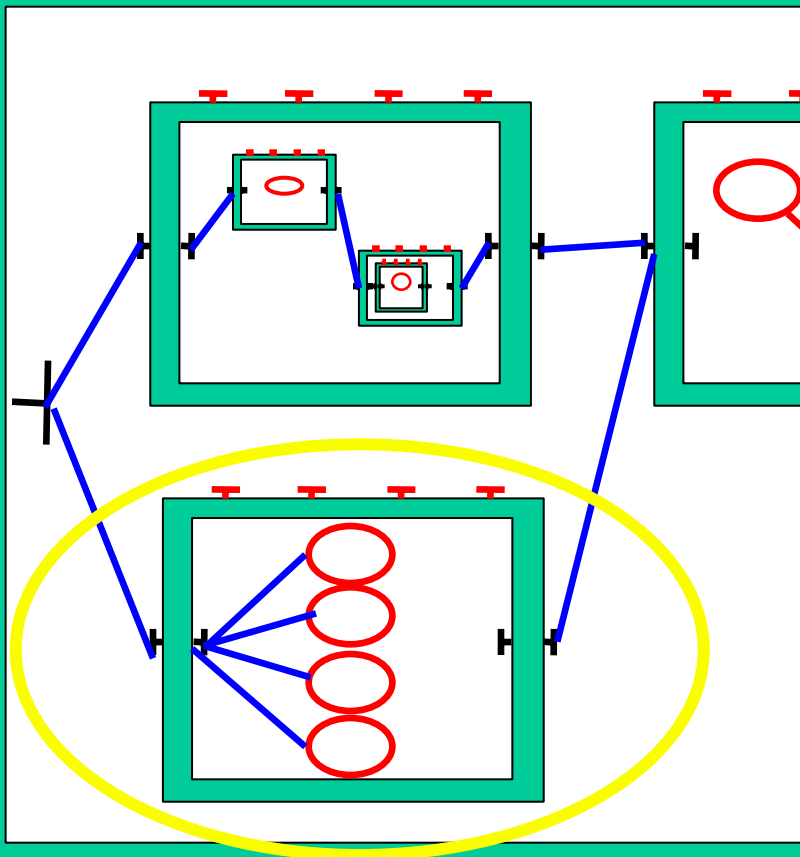
Messages



Distributed Components

Graphical Composition, Monitoring, Migration

Composition View



Monitoring Look & feel Window Globus

World Panel

The monitoring interface displays a network of VMs and their components. The components are organized into a hierarchy, with lines indicating connections between them. Key components highlighted with red circles include:

- globus4.inria.fr.Linux: C3DRenderingEngine#6
- A170.R2058.sc02.org.Linux: C3DRenderingEngine#7
- globus1.inria.fr.Linux: C3DRenderingEngine#5, C3DRenderingEngine#10
- globus3.inria.fr.Linux: C3DRenderingEngine#13, C3DRenderingEngine#9
- globus2.inria.fr.Linux: C3DRenderingEngine#18, C3DRenderingEngine#6
- pf12.inria.fr.Linux: C3DRenderingEngine#20, C3DRenderingEngine#21
- galere1.inria.fr.Linux: C3DRenderingEngine#15
- galere11.inria.fr.Linux: C3DRenderingEngine#17
- pf10.inria.fr.Linux: C3DRenderingEngine#8
- galere12.inria.fr.Linux: C3DRenderingEngine#22
- galere13.inria.fr.Linux: C3DRenderingEngine#24

Other components visible include GlobusRmi, GlobusJmi, C3DDispatcher, C3DUser, and various Renderers. The interface also includes a Messages section at the bottom with a 'clear messages' button.

Display topology proportional ratio filaire Monitoring enable



Adaptive Feature 2: Multi-transport layer RMI, RMI-ssh, ..., Ibis, HTTP XML, ...

Adaptive choice of transport layer between:

- RMI
- ssh/RMI

Also available with static configuration:

- Ibis (TCP, Myrinet, etc.)
- HTTP
- ... ssh/HTTP

Short Term Perspective:

Fully Adaptive Choice between all transports



2. *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 fulfilled)



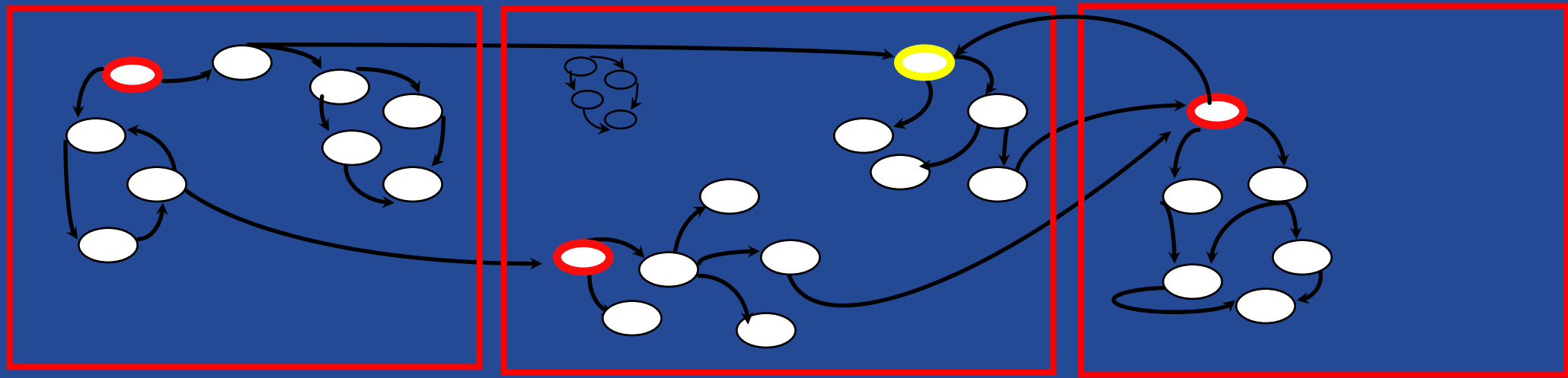
Principles and optimizations

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)



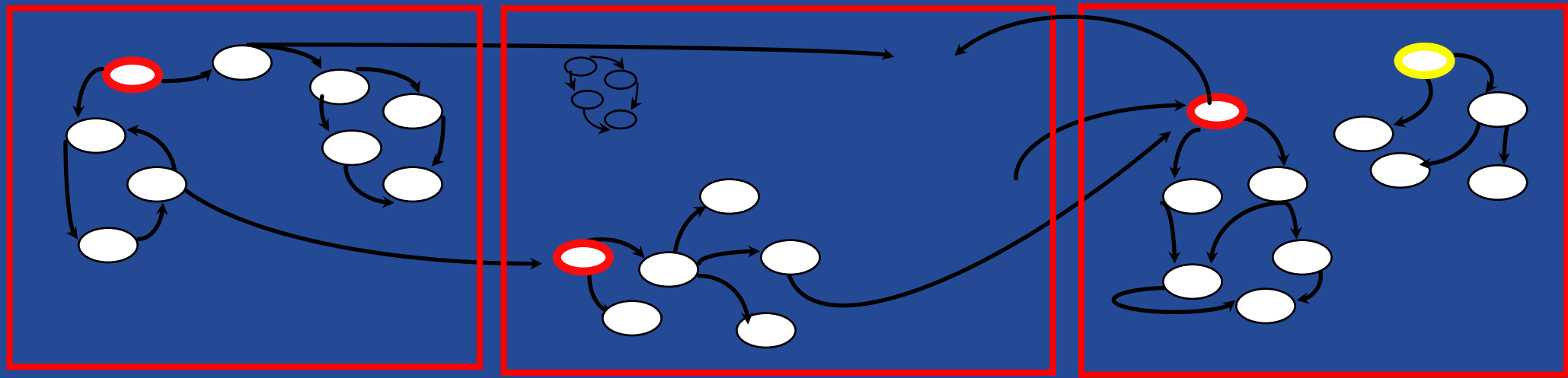
Principles and optimizations

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

Safe migration (no agent in the air!)

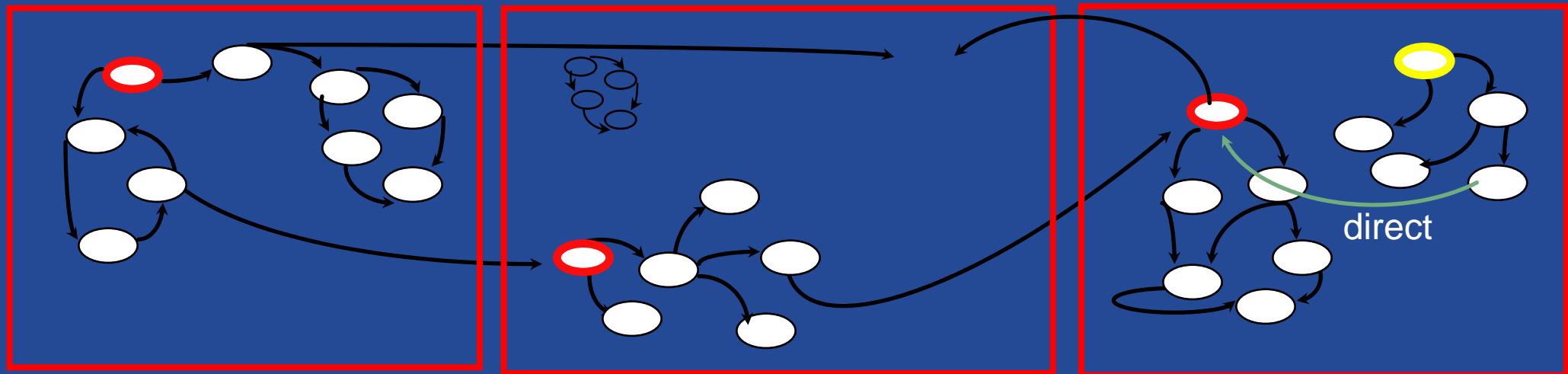
Local references if possible when arriving within a VM

Tensionning (removal of forwarder)



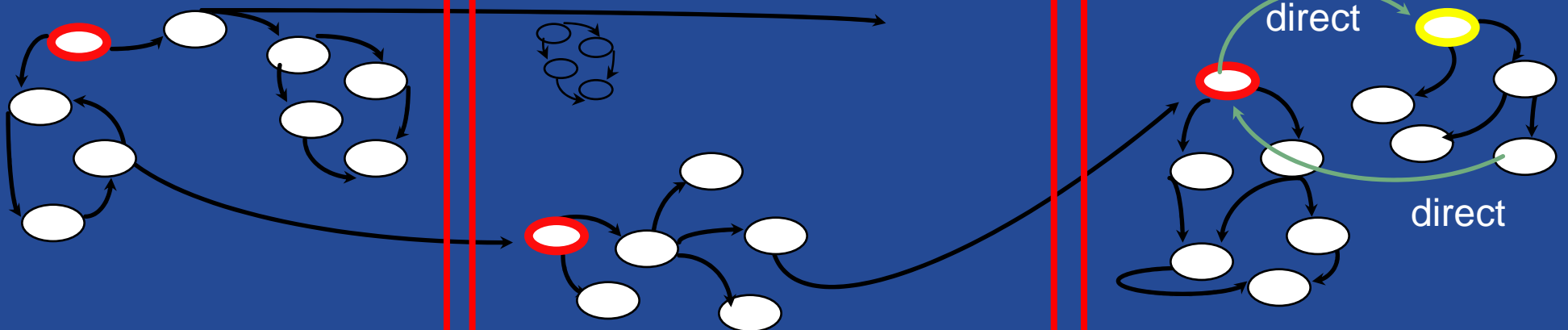
Principles and optimizations

- 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)



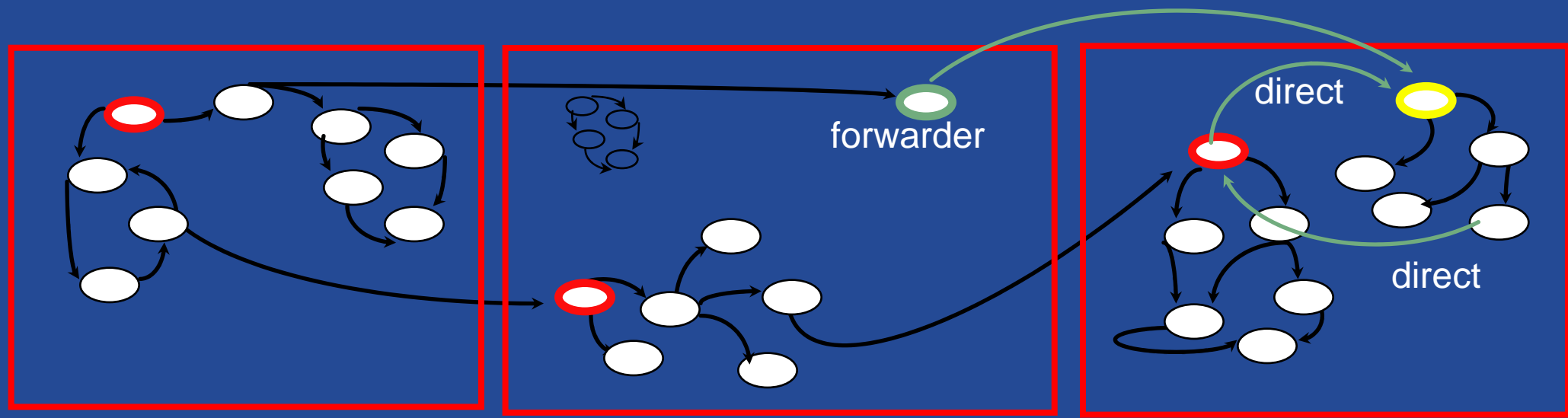
Principles and optimizations

- 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)



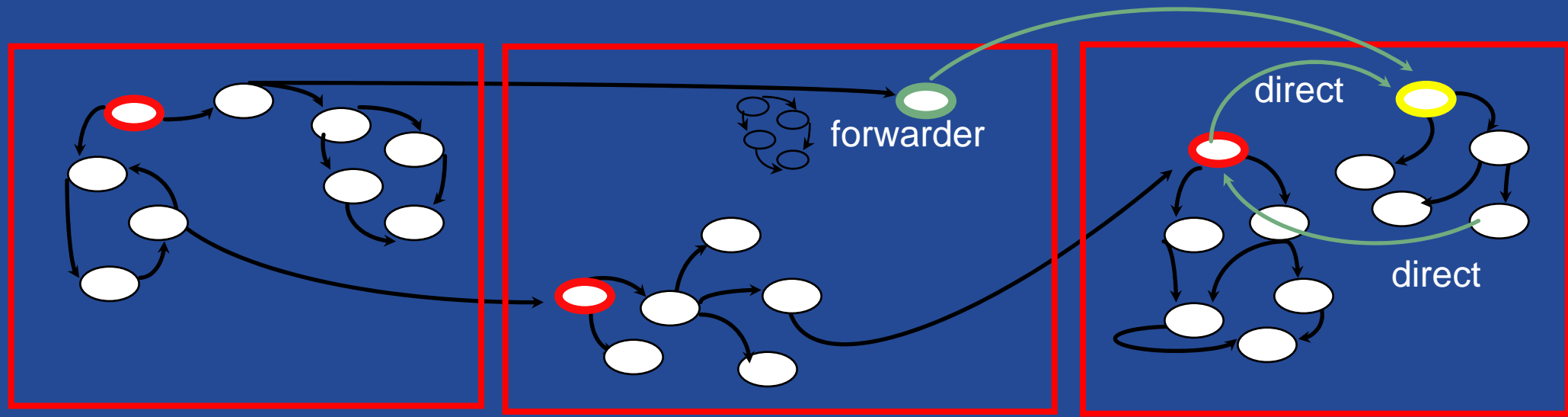
Principles and optimizations

- 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)



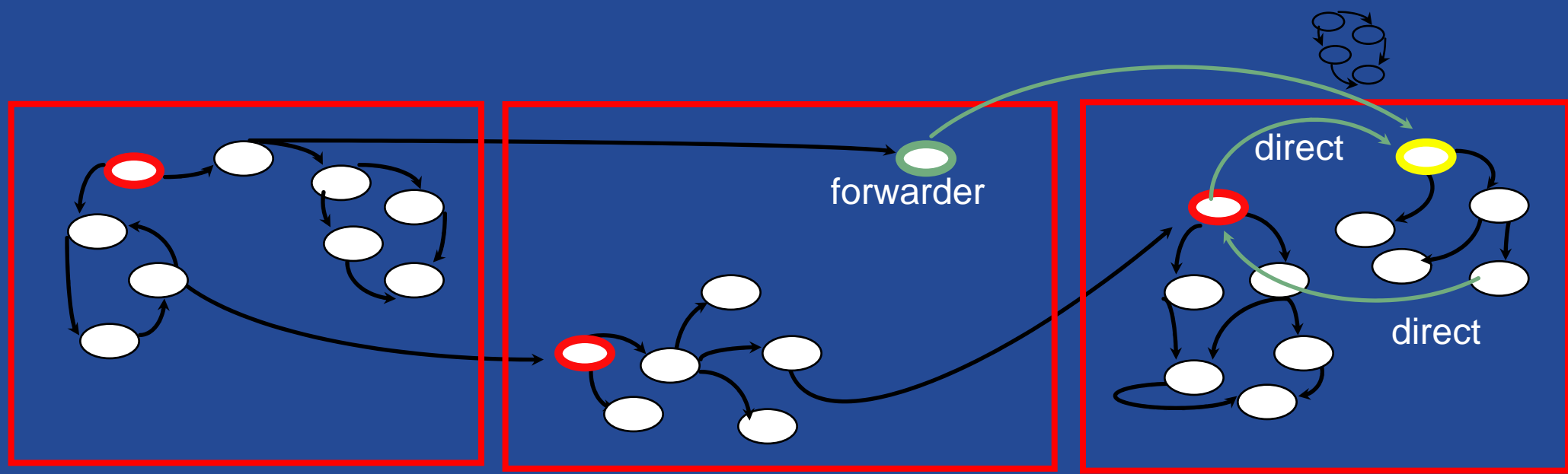
Principles and optimizations

- 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)



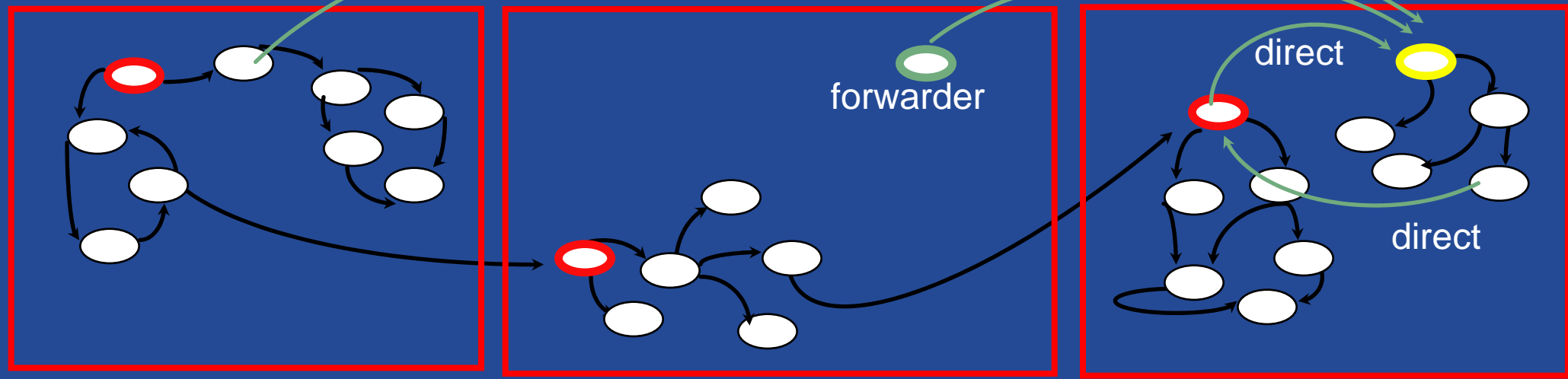
Principles and optimizations

- 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)



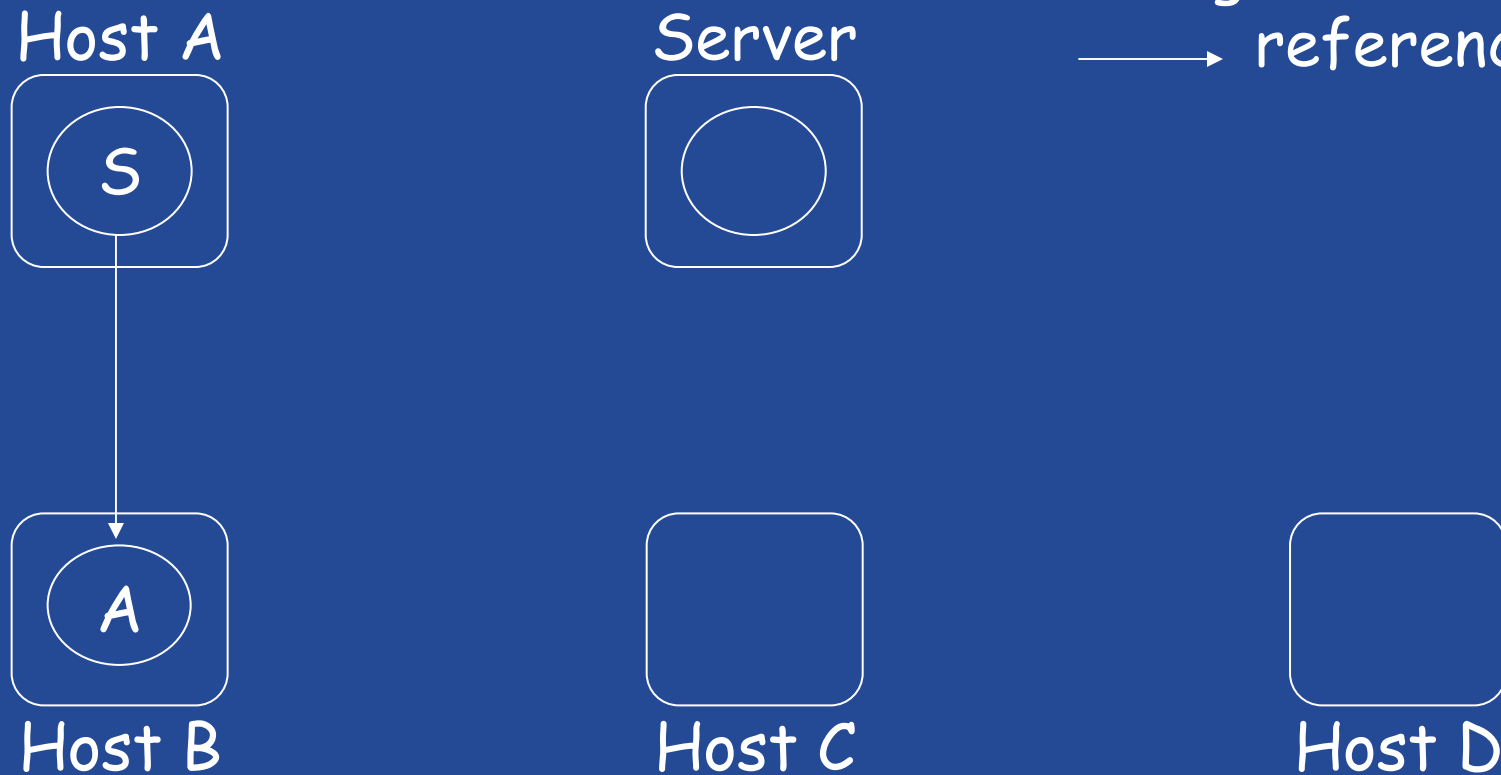
Principles and optimizations

- 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)



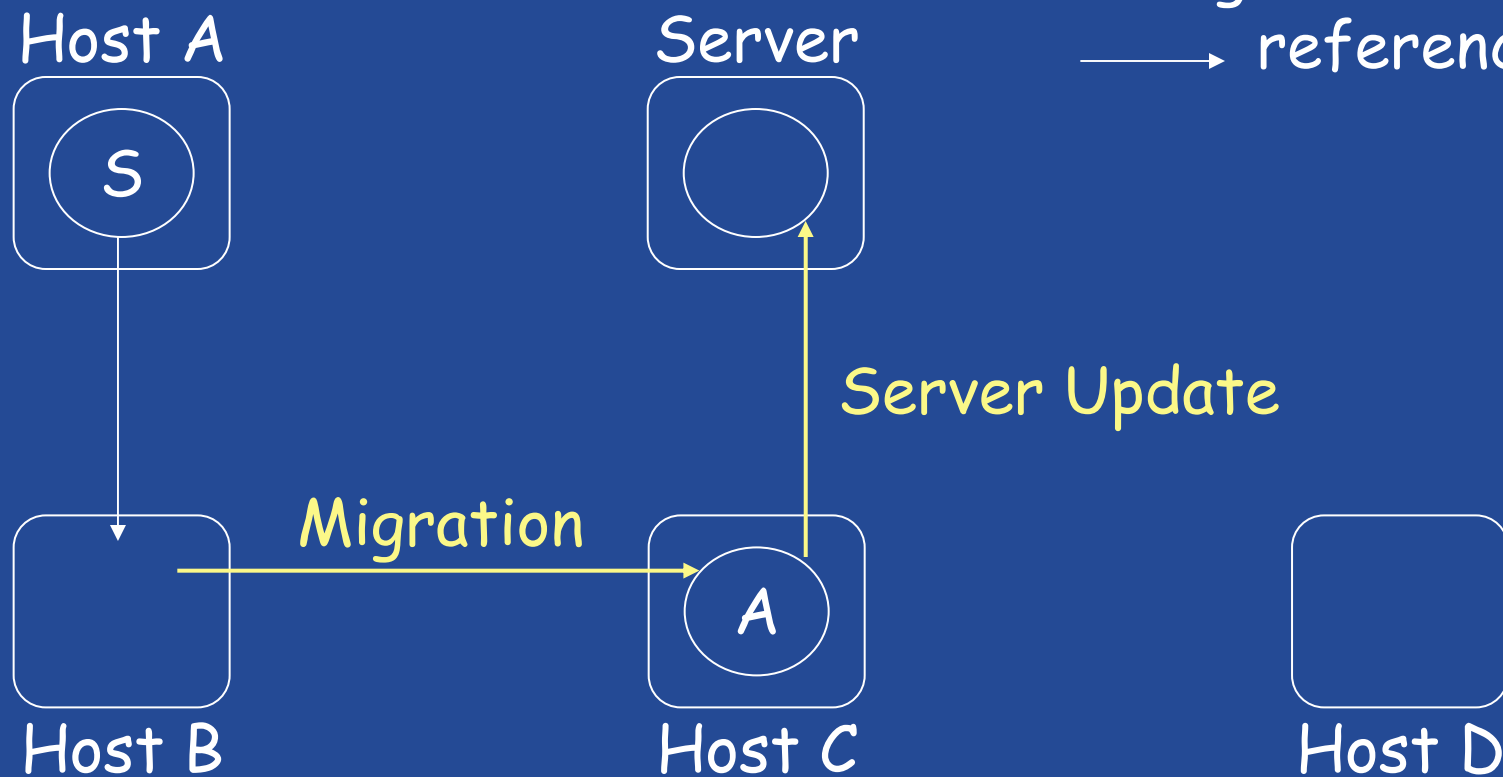
Other Strategy: Centralized

S : Source
A : Agent
→ reference



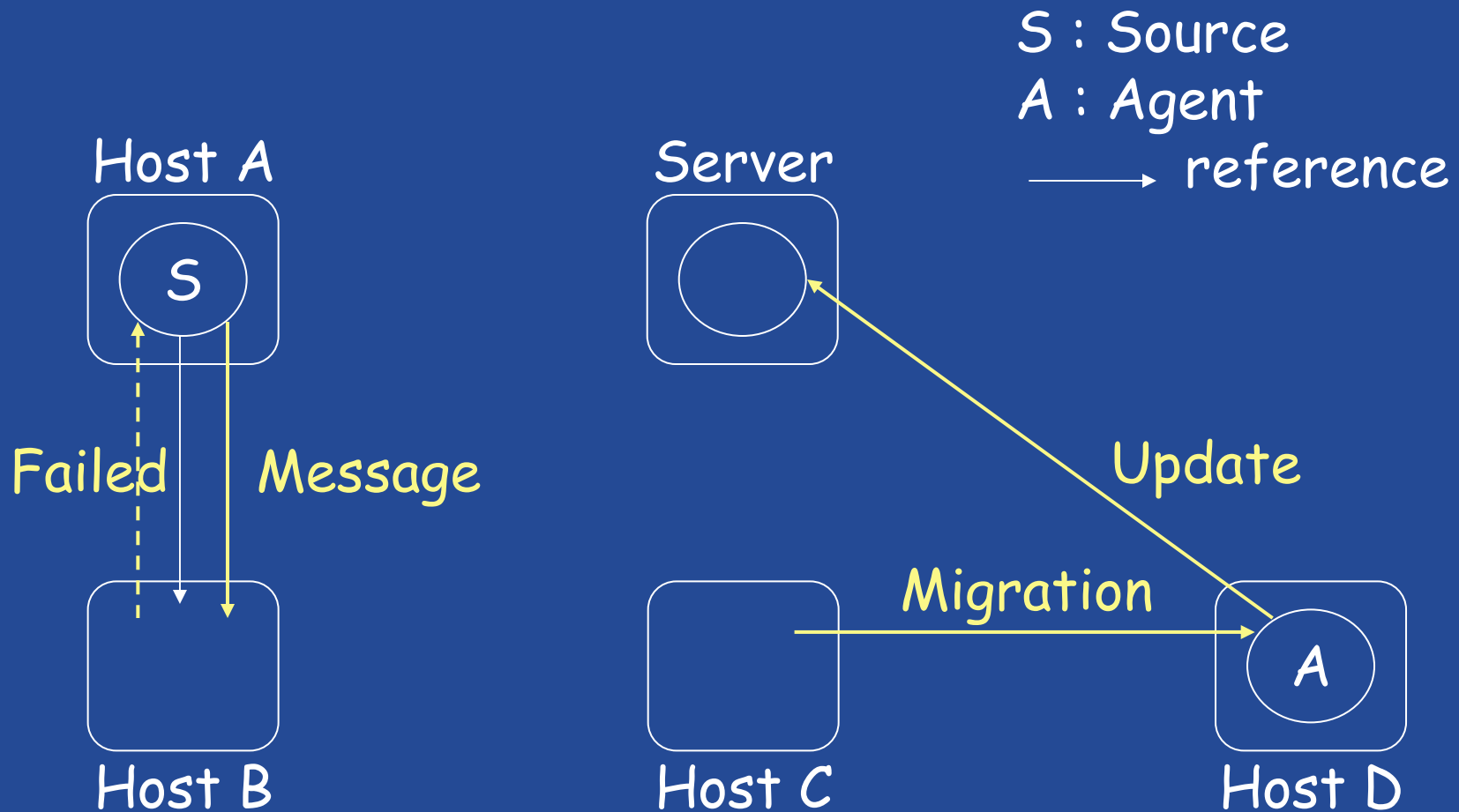
Centralized Strategy (2)

S : Source
A : Agent
→ reference



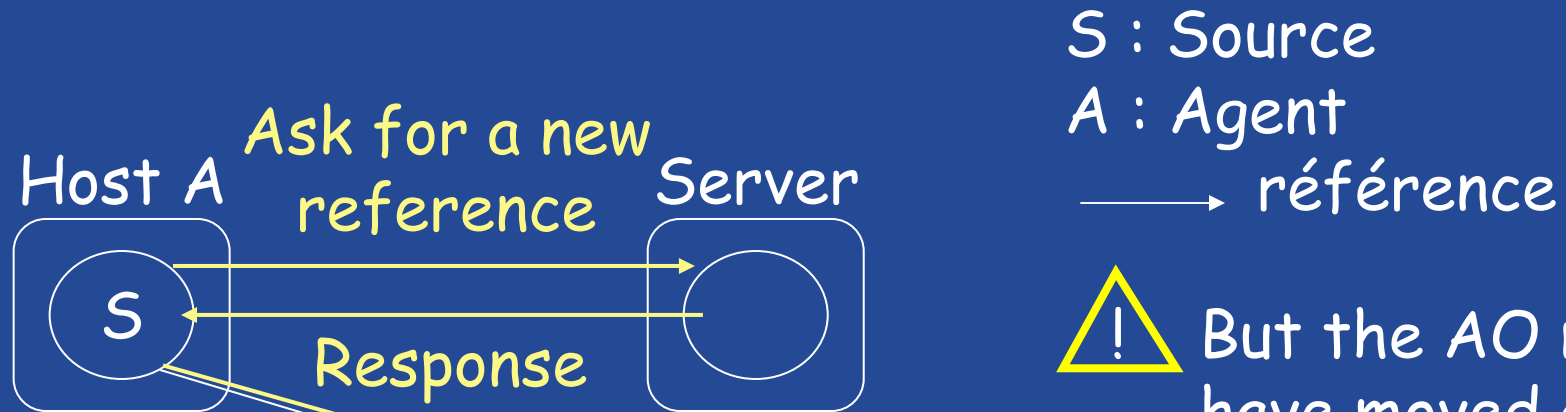
A migrating object updates the server

Centralized Strategy (3)

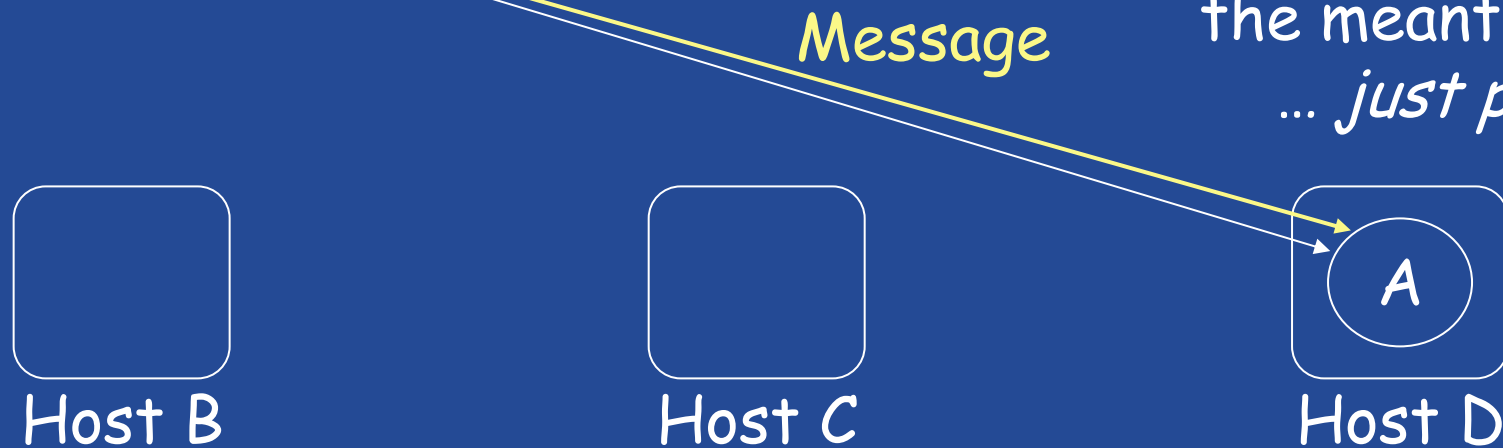


A migrating object updates the server

Centralized Strategy (4)



⚠ But the AO might have moved again in the meantime
... just play again.



The source get a new reference from the server

Adaptive Feature 3: TTL-TTU mixed parameterized protocol

TTL: Time To Live + Updating Forwarder:

5 s

- After TTL, a forwarder is subject to self destruction
- Before terminating, it updates server(s) with last agent known location

TTU: Time To Update mobile AO:

- After TTU, an will inform a localization server(s) of its current location

Dual TTU: first of two events:

- maxMigrationNb: the number of migrations without server update 10
- maxTimeOnSite: the time already spent on the current site 5 s



Adaptative Security



ProActive Security: Key Features

ProActive Security Features

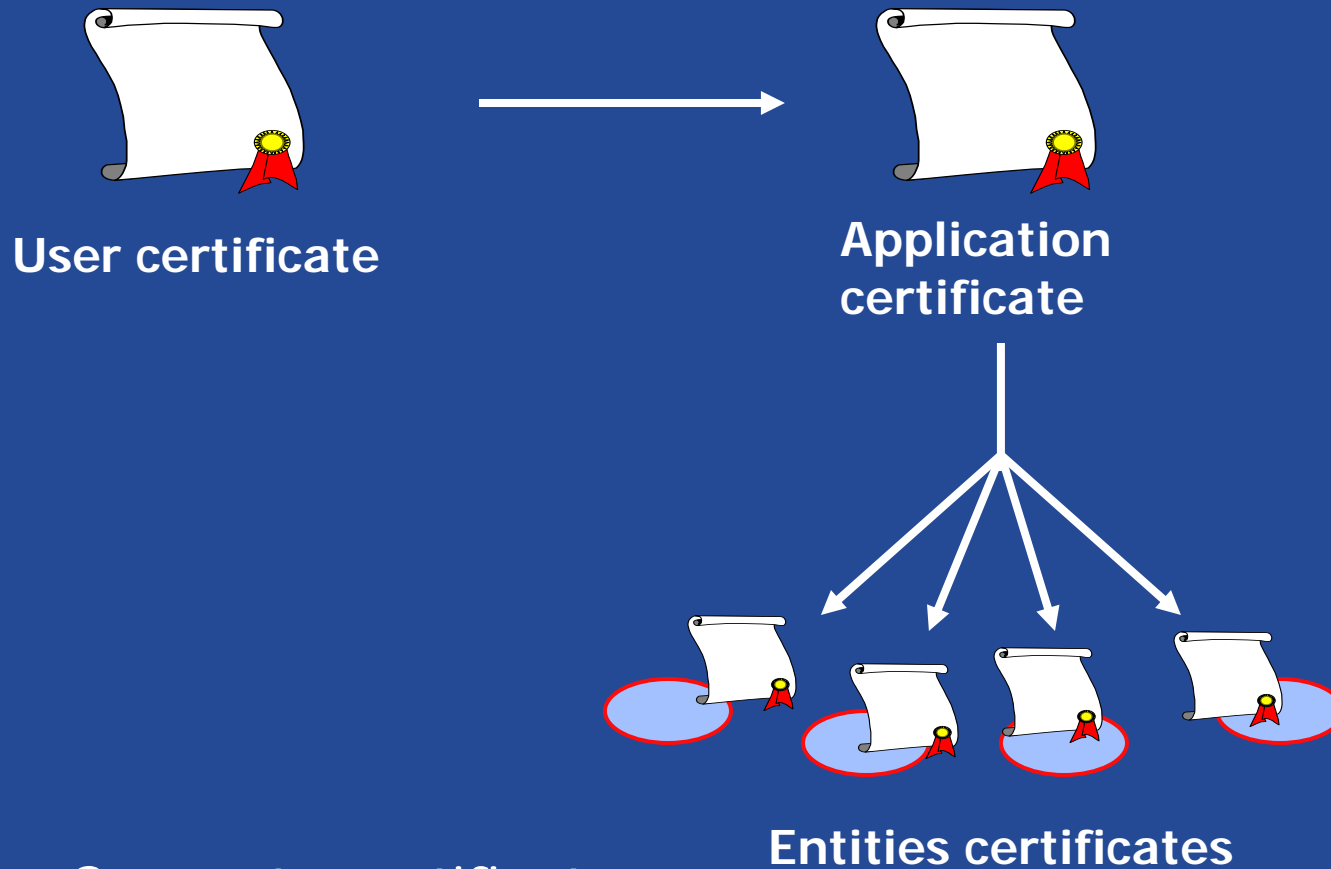
- Authentication of users and applications (PKI X 509 certificates)
- Authentication, Integrity and Confidentiality of communications

[A,I,C]

- In XML deployment files, Not In Source
- Mobility Aware
- Dynamically negotiated policies



A Chain of X509 Certificates

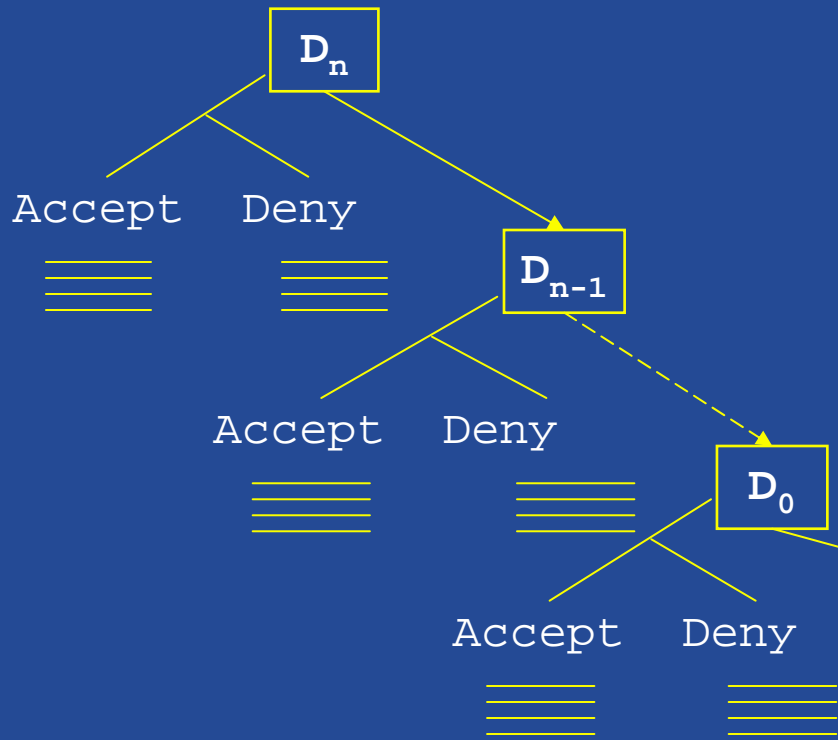


→ Generate certificate



Multi-level Policies

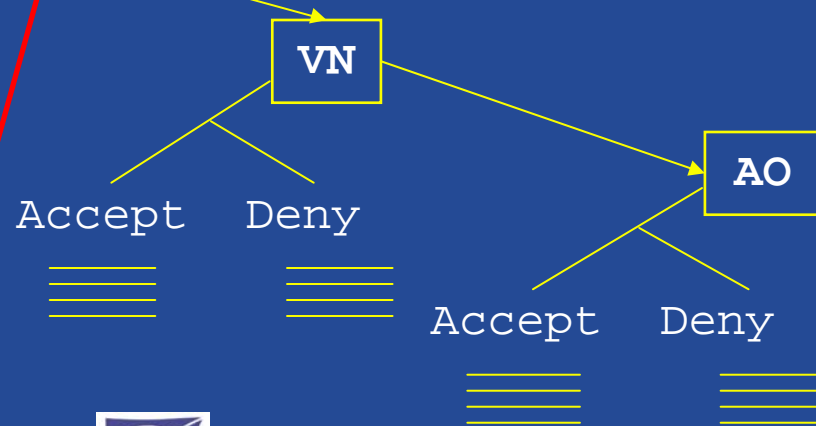
Administrator +
User-level policy



Security policy is defined according all matching rules from:

- Domains
- Virtual Nodes
- Active Objects

Application-level
policy



Security
policy



Combining Policies

Search for the most specific rule in each domain.

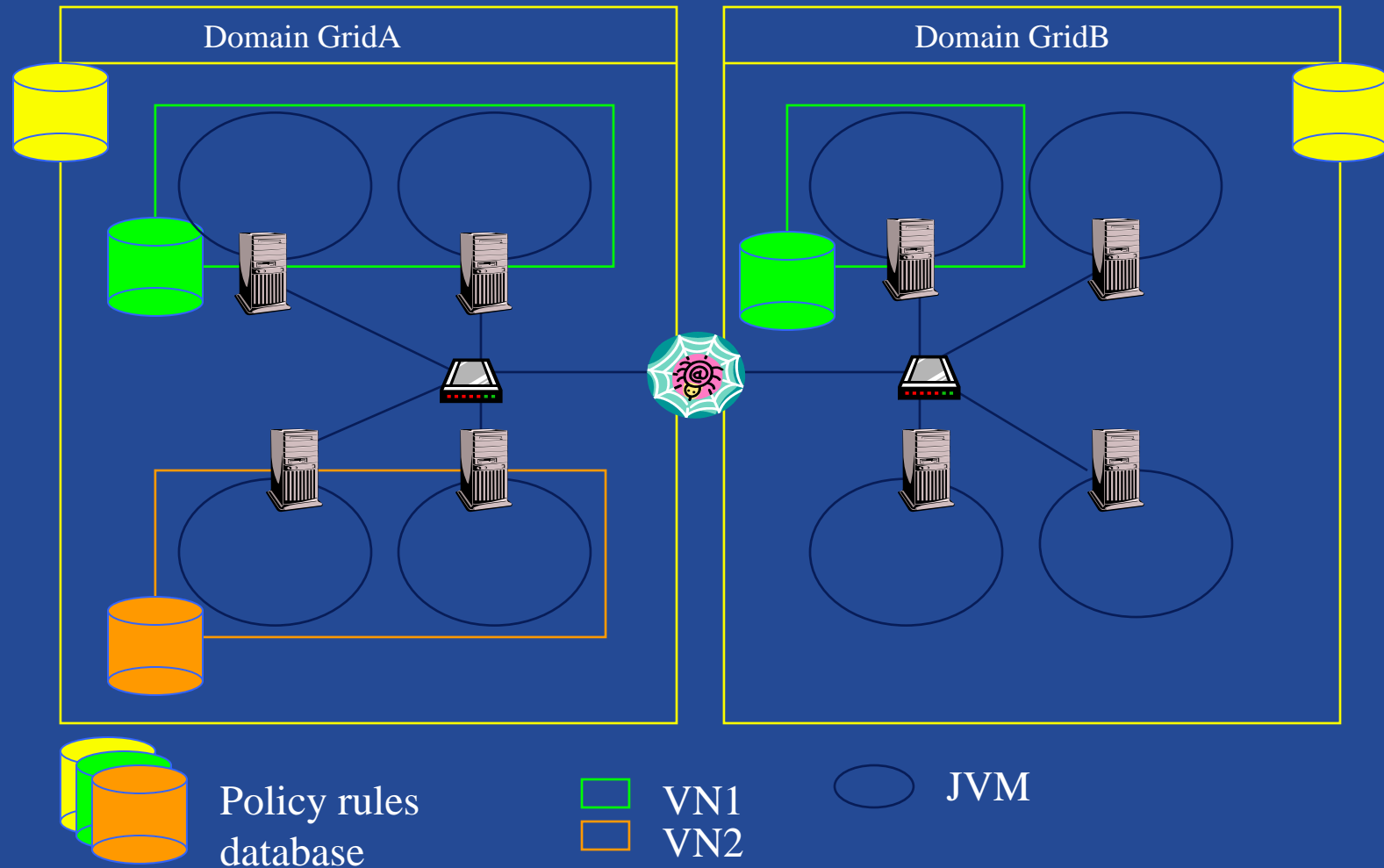
Retrieve all matching rules

Compute policies according to security attributes

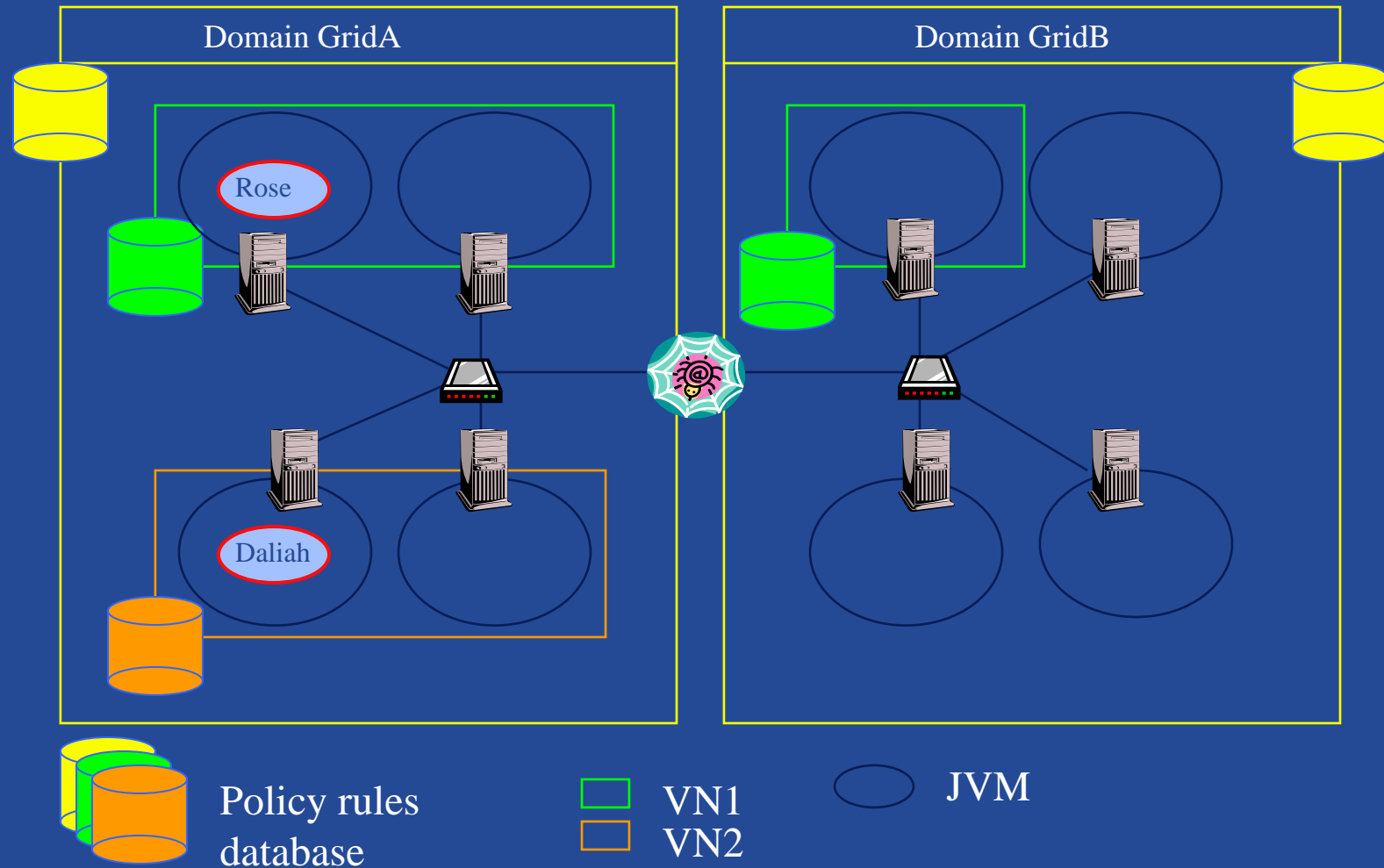
Receiver Sender	Required (+)	Optional (?)	Disallowed (-)
Required (+)	+	+	invalid
Optional (?)	+	?	-
Disallowed (-)	invalid	-	-



Example

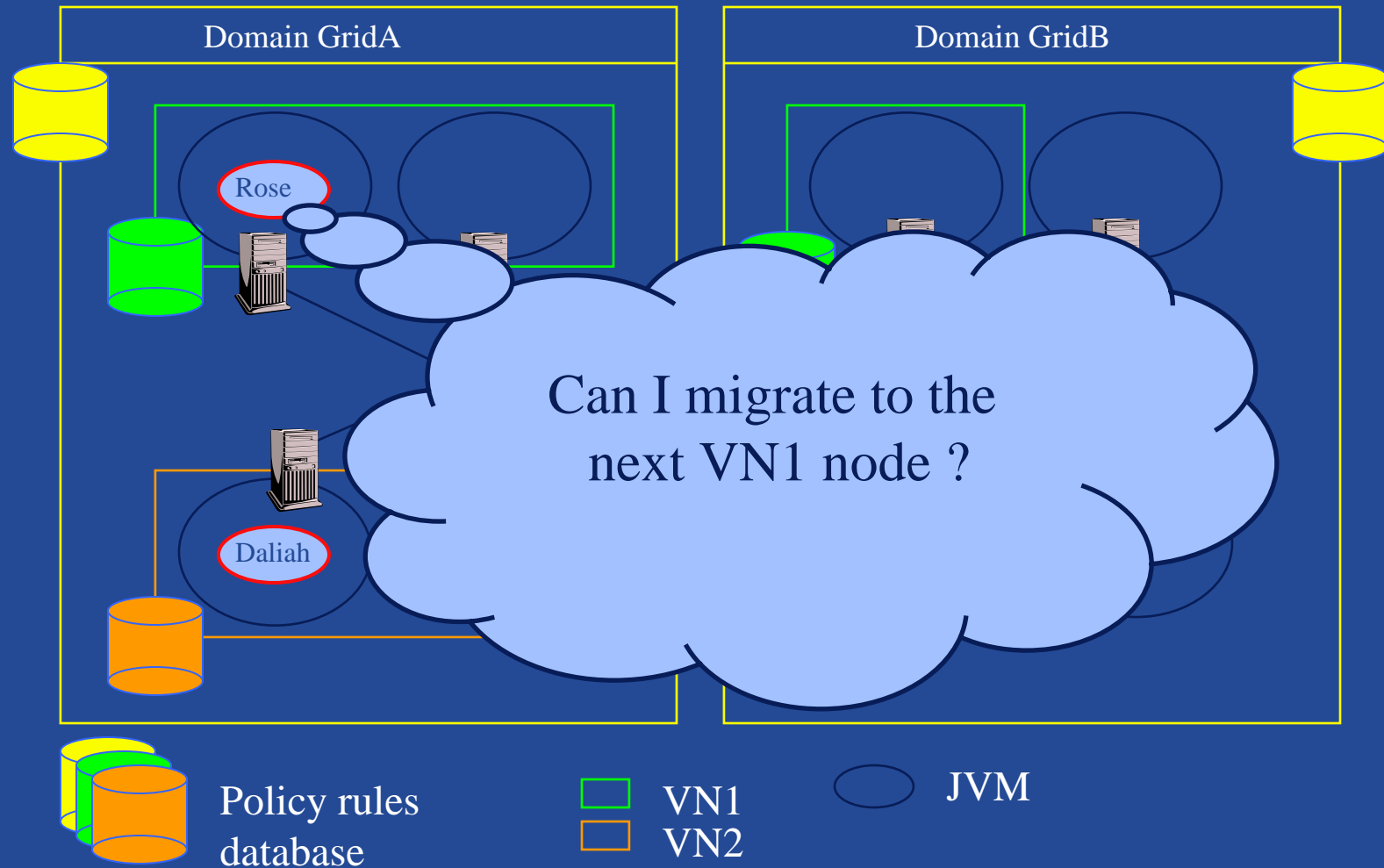


Example



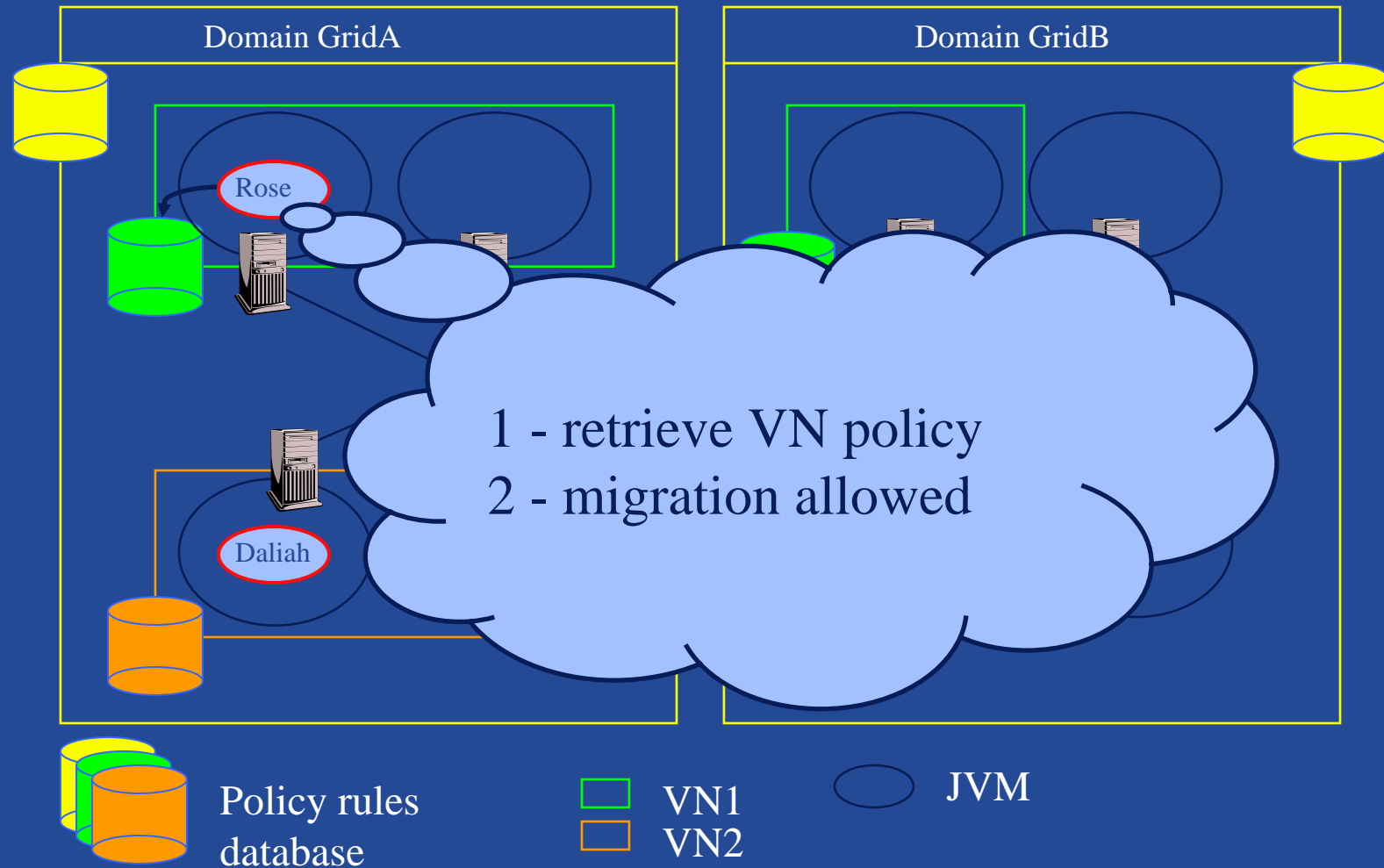
Example

- Migration :
- same VN
 - same domain



Example

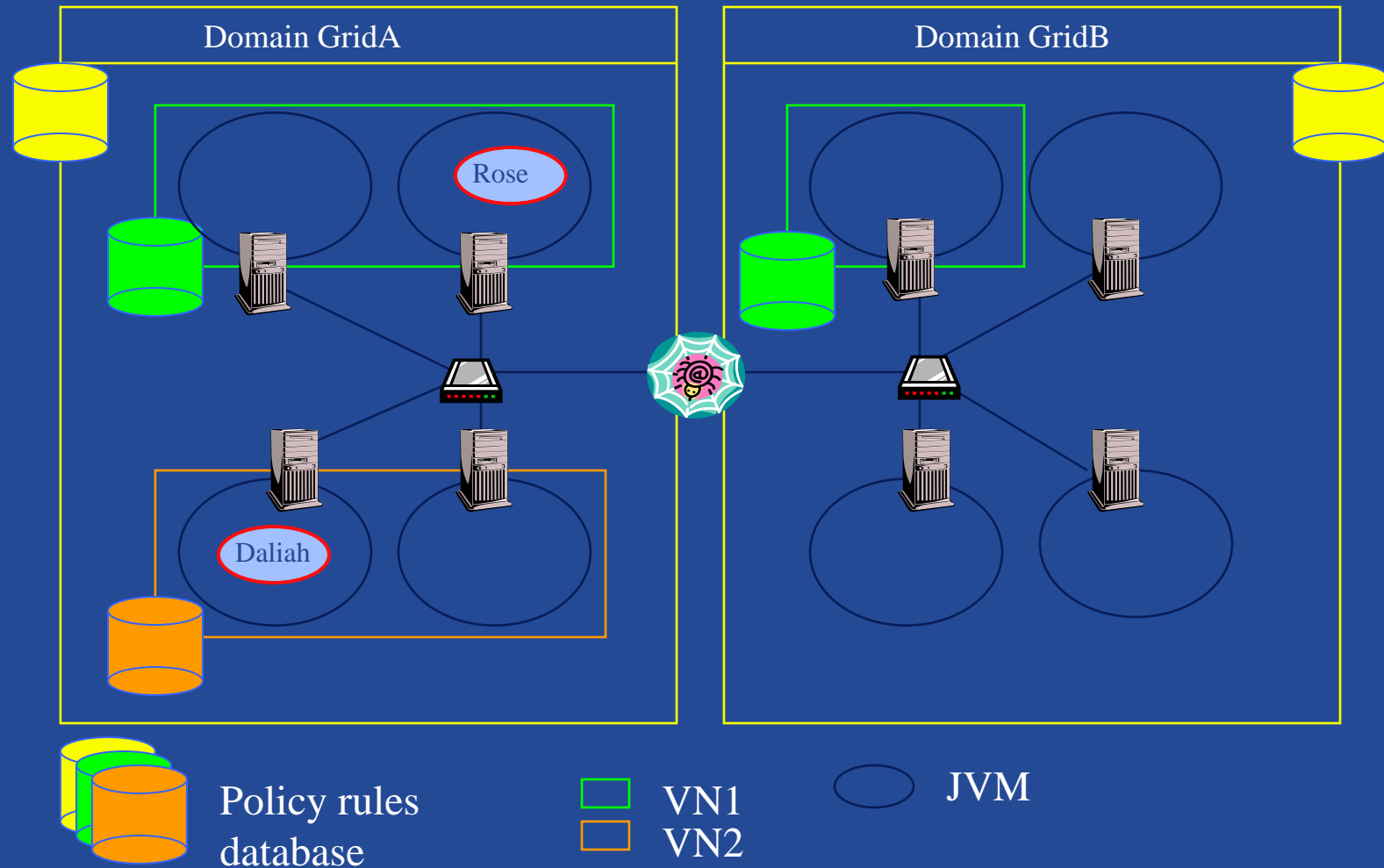
- Migration :
- same VN
 - same domain



Example

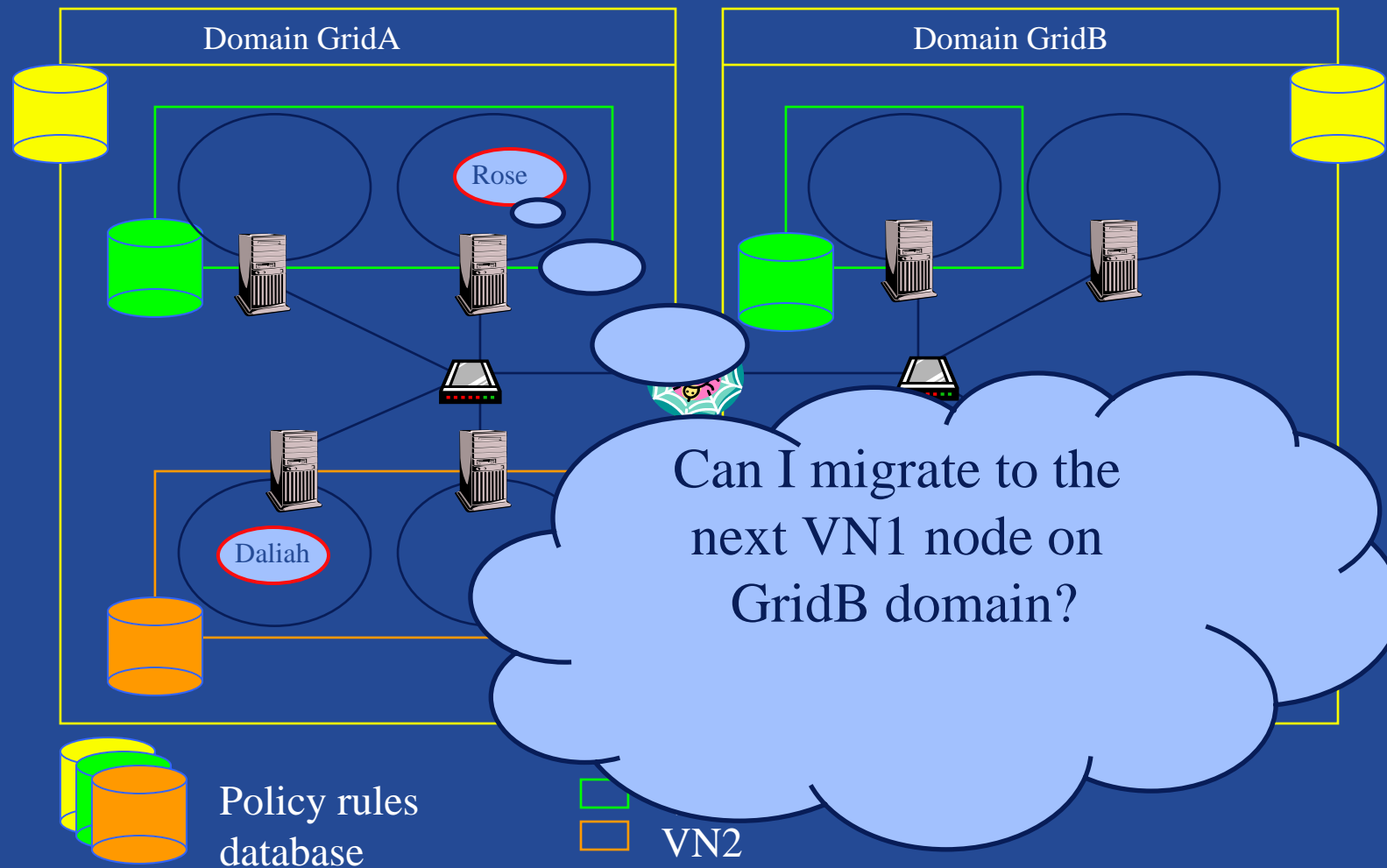
Migration :

- same VN
- same domain



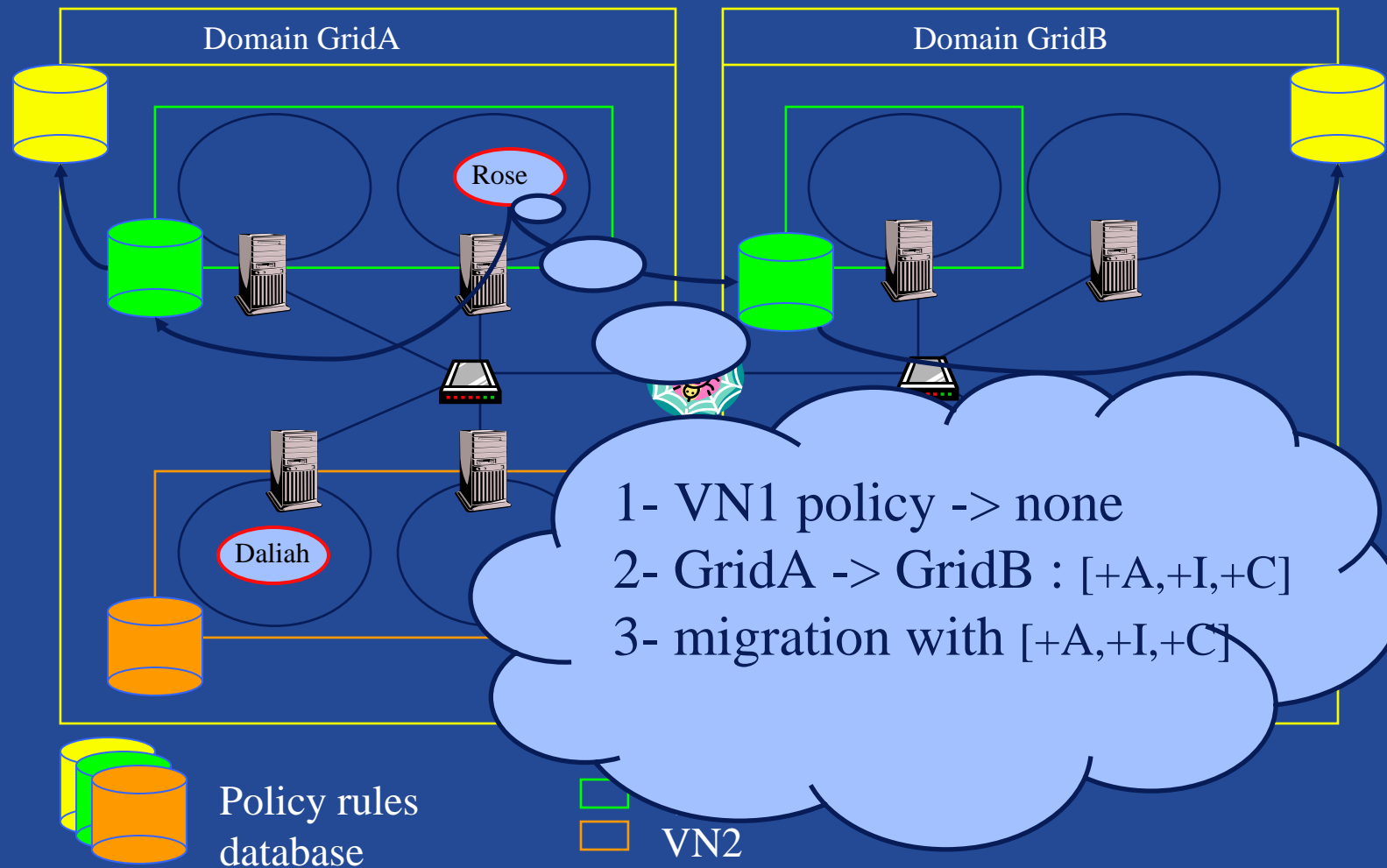
Example

Migration :
- same VN
- other domain



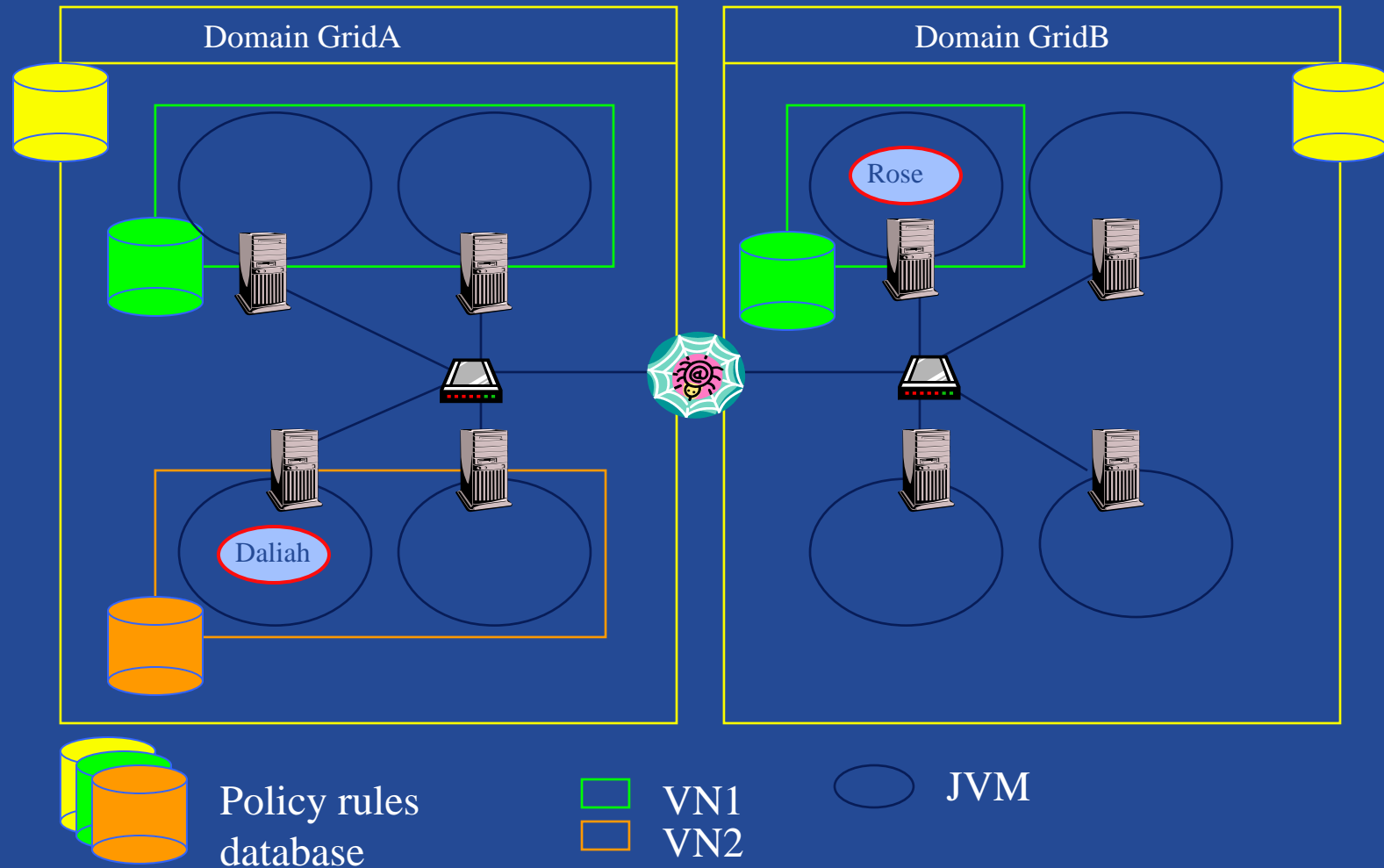
Example

Migration :
- same VN
- other domain



Example

Migration :
- same VN
- other domain

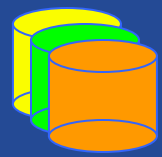
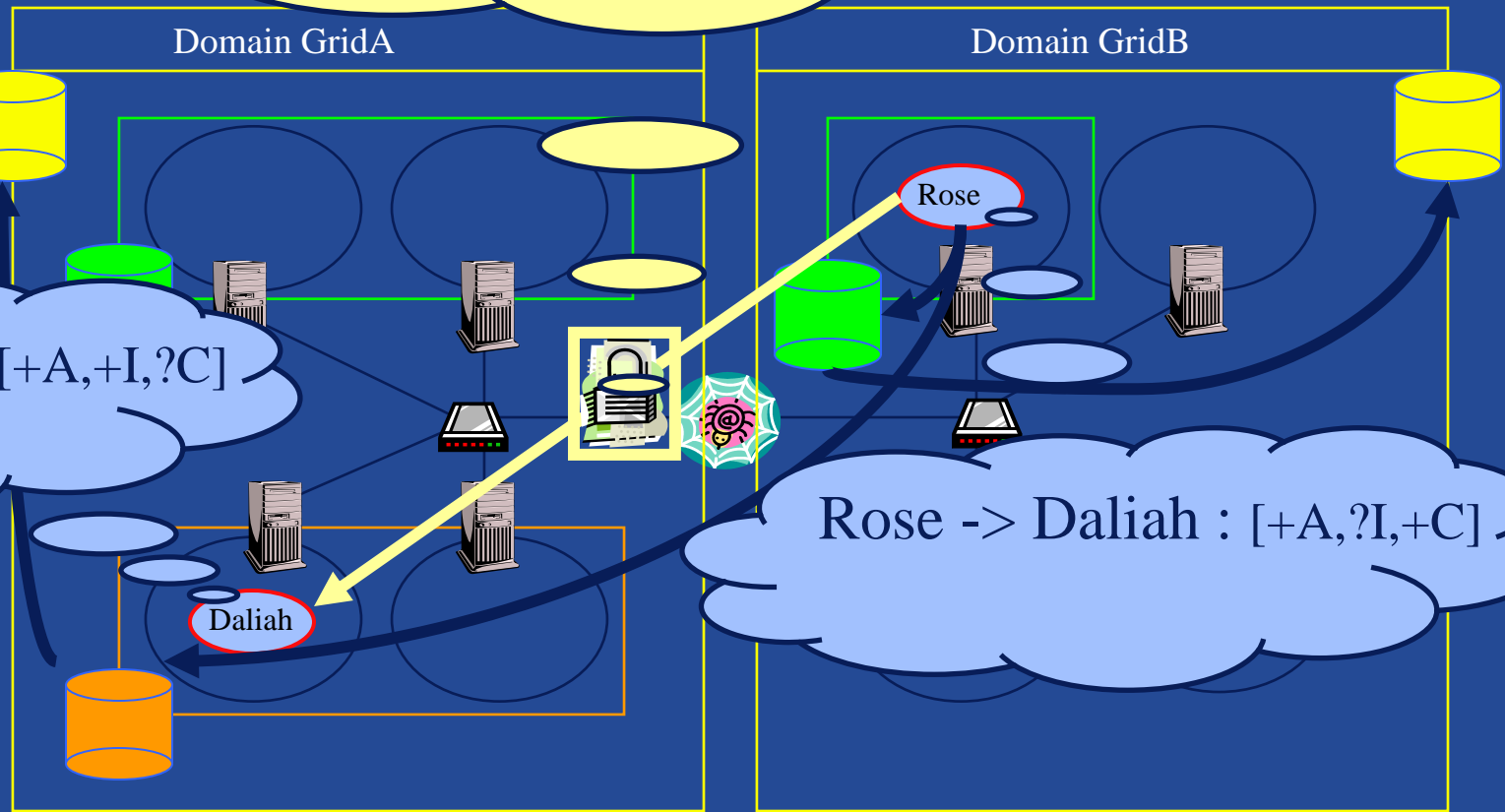


Negotiated Policy:
 Rose -> Daliah : [+A,+I,+C]

Method call :
 - other VN
 - other domain
 From
 Rose --> Daliah

Daliah -> Rose : [+A,+I,?C]

Rose -> Daliah : [+A,?I,+C]



Policy rules
 database

VN1
 VN2

JVM



Adaptive Feature 4: Adaptive Security

Dynamic setting of the security attributes

Dynamic negotiation between different:

- Domain and Sub-domain
- Virtual Nodes
- Active Objects

on JVMs on different Machines



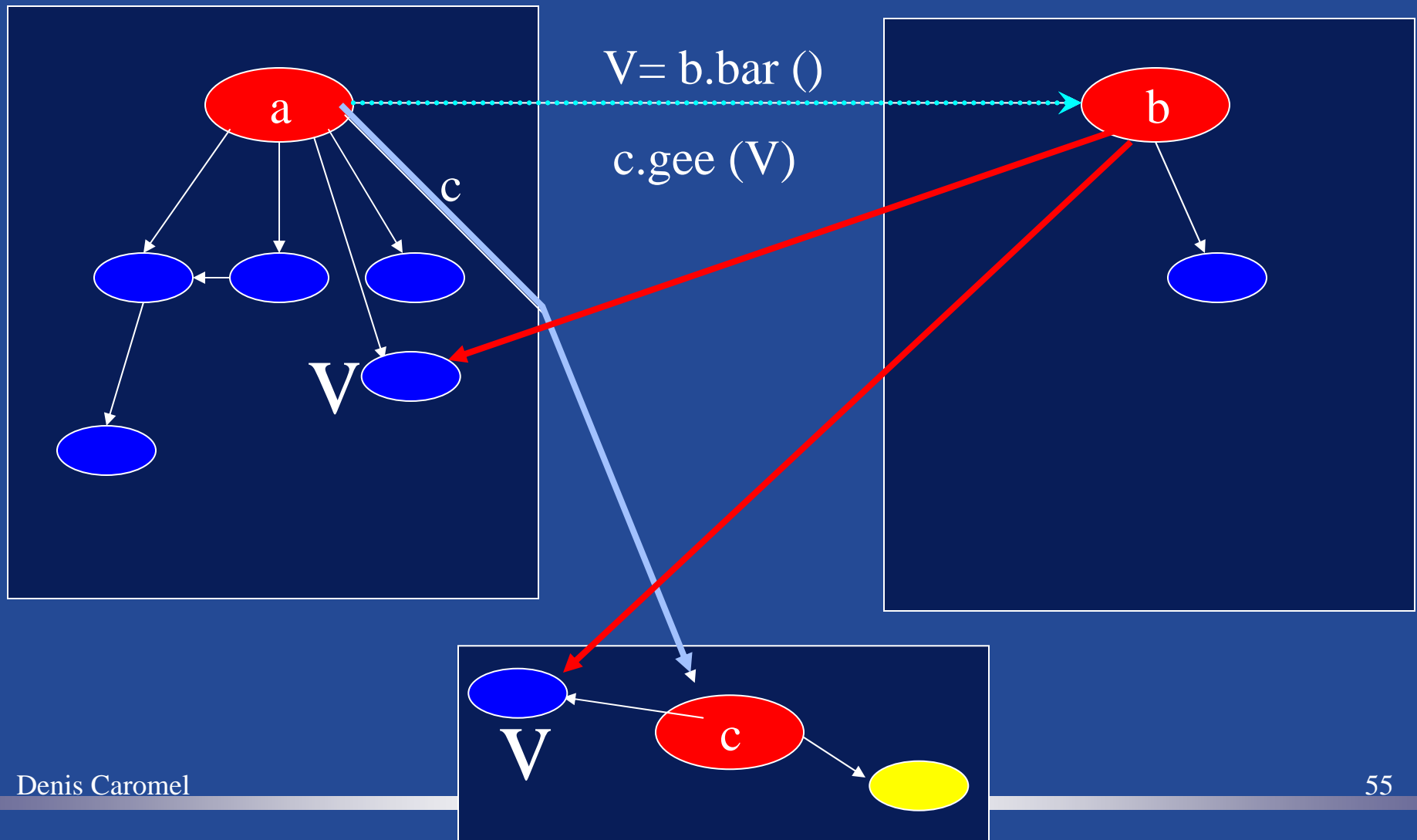
First-Class Futures

Update



Wait-By-Necessity: First Class Futures

Futures are Global Single-Assignment Variables



Adaptive Feature 5: Future update strategies

No partial replies and requests:

- No passing of futures between activities, more deadlocks

Eager strategies: as soon as a future is computed

- Forward-based:
 - Each activity is responsible for updating the values of futures it has forwarded
- Message-based:
 - Each forwarding of future generates a message sent to the computing activity
 - The computing activity is responsible for sending the value to all

Mixed strategy:

- Futures update any time between future computation and WbN

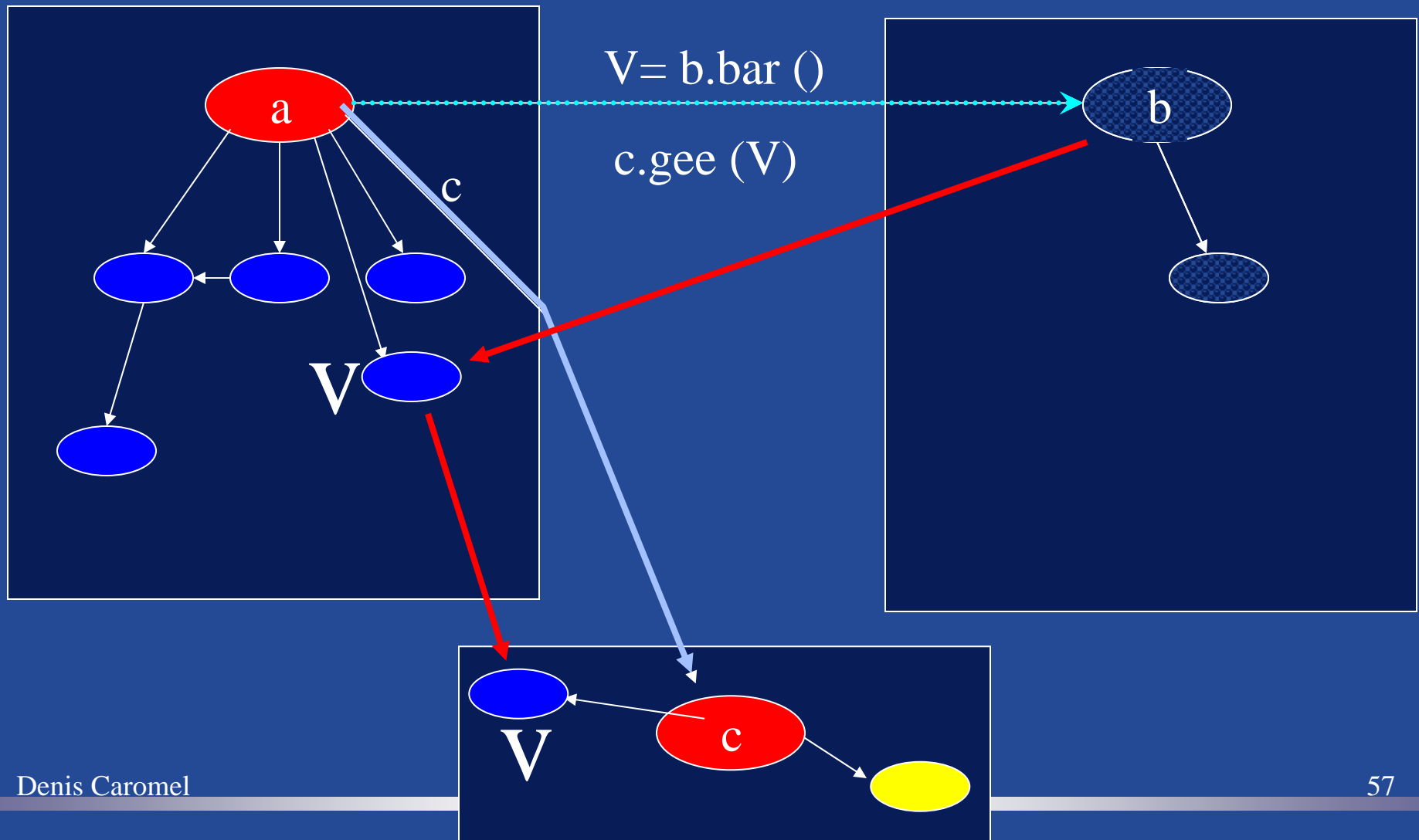
Lazy strategy:

- On demand, only when the value of the future is needed (WbN on it)



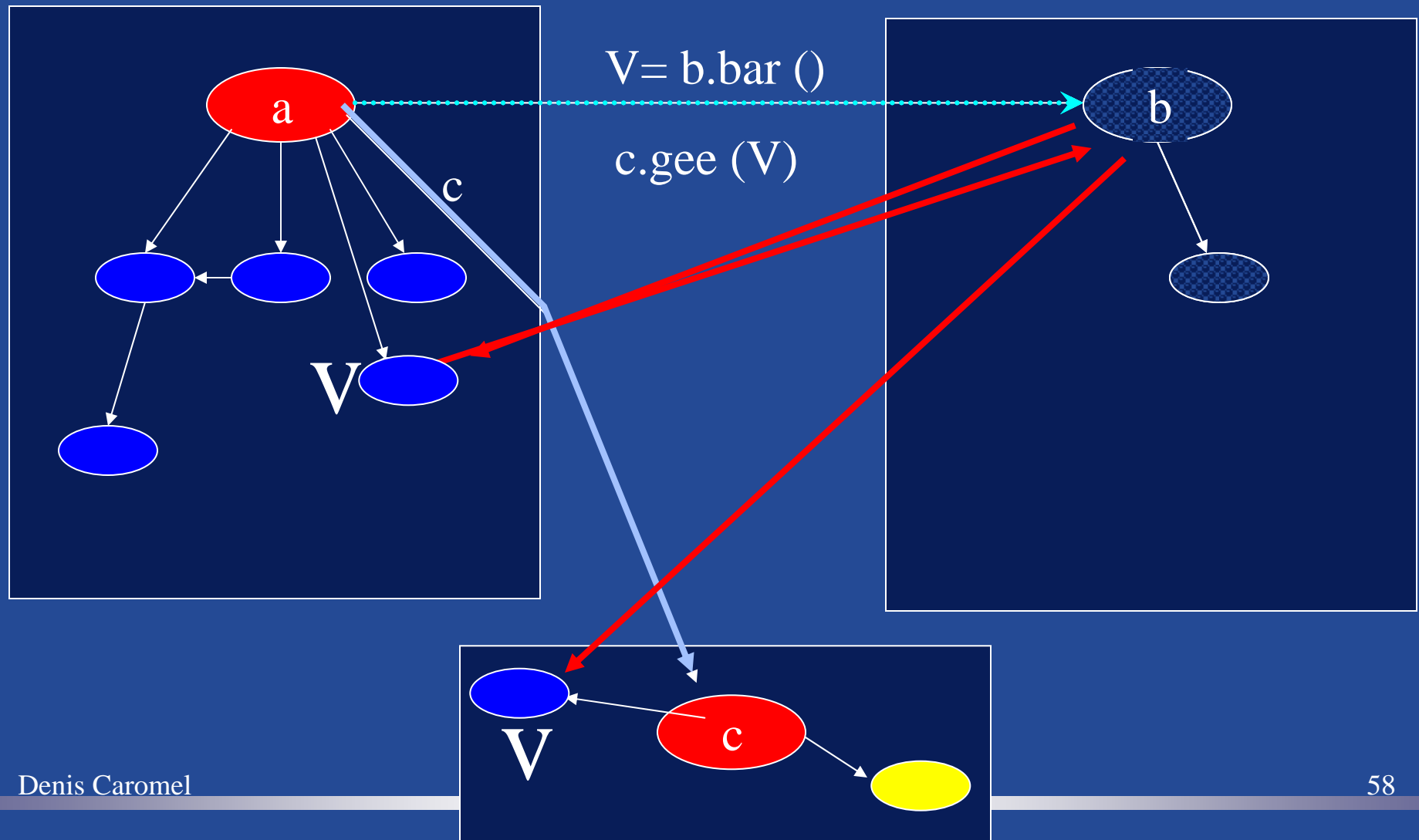
Wait-By-Necessity: Eager Forward Based

AO forwarding a future: will have to forward its value



Wait-By-Necessity: Eager Message Based

AO forwarding a future: send a message



Adaptive: Active Objects, Cp. vs. MPI



MPI Communication primitives

For some (historical) reasons, MPI has many com. Primitives:

MPI_Send	Std	MPI_Recv	Receive
MPI_Ssend	Synchronous	MPI_Irecv	Immediate
MPI_Bsend	Buffer	... (any) source, (any) tag,	
MPI_Rsend	Ready		
MPI_Isend	Immediate, async/future		
MPI_Ibsend, ...			

First of all:

- a combinatory complexity occurs between sendS and receiveS
- many semantic variation and problems arise between implementations

I'd rather put the burden on the implementation, not the Programmers !

How to do adaptive implementation in that context ?

Is Recv at all needed ? First adaptive feature: Dynamic Control Flow of Mess.



Main MPI problems for the GRID

Too static in design

Too complex in Interface (API)

Too many specific primitives to be adaptive

Typelessness



Sum up: MPI vs. ProActive / OO SPMD

A simple communication model, with simple communication primitive(s):

- No RECEIVE but data flow synchronization
- Adaptive implementations are possible for:
 - // machines, Cluster, Desktop, etc.,
 - Physical network, LAN, WAN, and network conditions

Typed Method Calls:

==> Component enabled

... Adaptivity is needed for Components



Adaptive GRID

The need for adaptive middleware is now acknowledged, with dynamic strategies at various points in containers, proxies, etc.

Can we afford adaptive GRID ?

with dynamic strategies at various points

... communications, groups, checkpointing, reconfiguration, ...
to deal with various conditions (LAN, WAN, network, P2P, ...)

YES !

HPC vs. HPC

High Performance Components vs. High Productivity Components



Conclusion

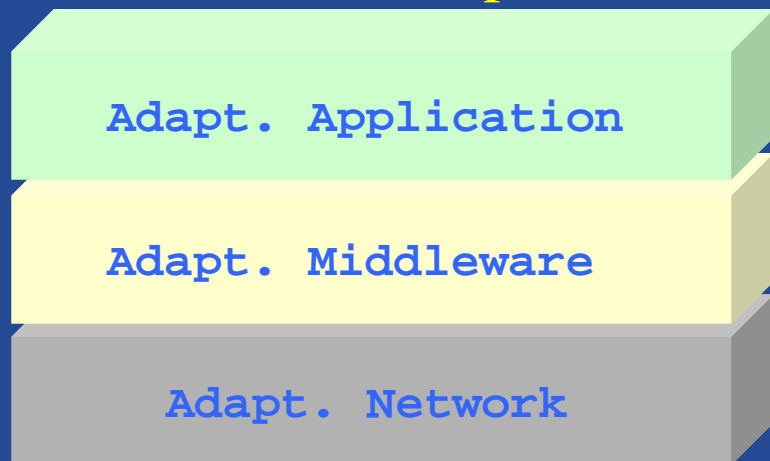
5 Adaptive/Parameterized features in *ProActive*:

- RMI <--> ssh/RMI ... HTTP - Ibis/TCP - Ibis/Myrinet - ...
- Groups, Localization in Mobility, Security, Future Update

Perspectives -- On-going work:

- Adaptive Components:
 - Tensionning
 - Re-configuration
- Adaptive Checkpointing

Better off with simple Functional RMI / Two-sided MPI Message Passing



Lets just be careful:
otherwise, we'll just build ...

Maladaptive Adaptive Grids !

TCP is an Adaptive Middleware



ProActive.ObjectWeb.org



- [Home](#)
- [Documentation - API - Papers](#)
- [IC2D](#)
- [Applications](#)
- [Download](#)
- [What's new?](#)
- [CVS Access](#)
- [FAQ](#)
- [Mailing list](#)
- [Users](#)
- [Links](#)
- [Bibtex](#)
- [Feedback](#)
- [Contacts](#)
- [Jobs](#)

ProActive

Programming, Composing, Deploying on the Grid



New version 2.0 with source code (April 2004) IS HERE

GRID PLUGTESTS, CONTEST, and ProActive USER GROUP organized by ETSI, Oct. 18-20

Solve the NQueen problem with ProActive

ProActive Tutorial+Hands-on @ Euro-Par 2004 Pisa

ProActive is a Java library ([Source code under LGPL licence](#)) for **parallel, distributed, and concurrent** computing, also featuring **mobility and security** in a uniform framework. With a reduced set of simple primitives, **ProActive** provides a comprehensive API allowing to simplify the programming of applications that are distributed on **Local Area Network (LAN)**, on **cluster of workstations**, or on **Internet Grids**.

The library is based on an **Active Object pattern** that is a uniform way to encapsulate:

- a **remotely** accessible object,
- a **thread** as an asynchronous activity,
- an **actor** with its own script,
- a **server** of incoming requests,
- a **mobile** and potentially secure agent.

ProActive is only made of standard Java classes, and requires **no changes to the Java Virtual Machine**, no preprocessing or compiler modification; programmers write standard Java code. Based on a simple Meta-Object Protocol, the library is itself extensible, making the system open for adaptations and optimizations. **ProActive** currently uses the **RMI** Java standard library as a portable transport layer.

A **graphical interface**, [IC2D](#), allows the remote monitoring and steering of distributed applications.

ProActive features the following:

- **Asynchronous calls**: Typed Messages (Request and Reply); **RMI + JMS** !
- Automatic **future-based** synchronizations: **wait-by-necessity**
- **Migration**, Mobile Agents (compatible with **Swing** and **AWT**)
- **Remote** creation of **remote objects**
- Reuse: **polymorphism** between standard objects and remote objects
- **Group Communications** with dynamic group management
- Libraries for sophisticated **synchronizations**, collaborative applications
- **Transparent, dynamic** code loading (up and down)
- **Seamless** management of the **RMIRegistry** and **Jini**
- **XML Deployment Descriptors**
- Interfaced with **Globus, Jini, LSF, PBS, rsh, ssh, rlogin, etc.**

New

- [ProActive Components](#) based on the **Fractal** model
- ProActive [Security](#) Framework
- [ProActive Reference Card](#): get all main concepts and methods at one glance
- XML [Configuration file](#) to enhance flexibility in ProActive
- [SAP experiments with ProActive](#)
- [ProActive wireless version for PDA](#)
- ProActive based Electromagnetism application [JEM3D](#) deployed on **294 processors** in P2P Desktop Grid
- [ProActive demonstrations at SC'03, Phoenix](#)



Version 2.0
April 2004

[Mailing list with public archive](#)

[Bug Tracking System](#)

[Mail to ProActive support](#)



The ProActive Team
Download ProActive 2.0
<http://www.inria.fr/oasis/ProActive>

