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

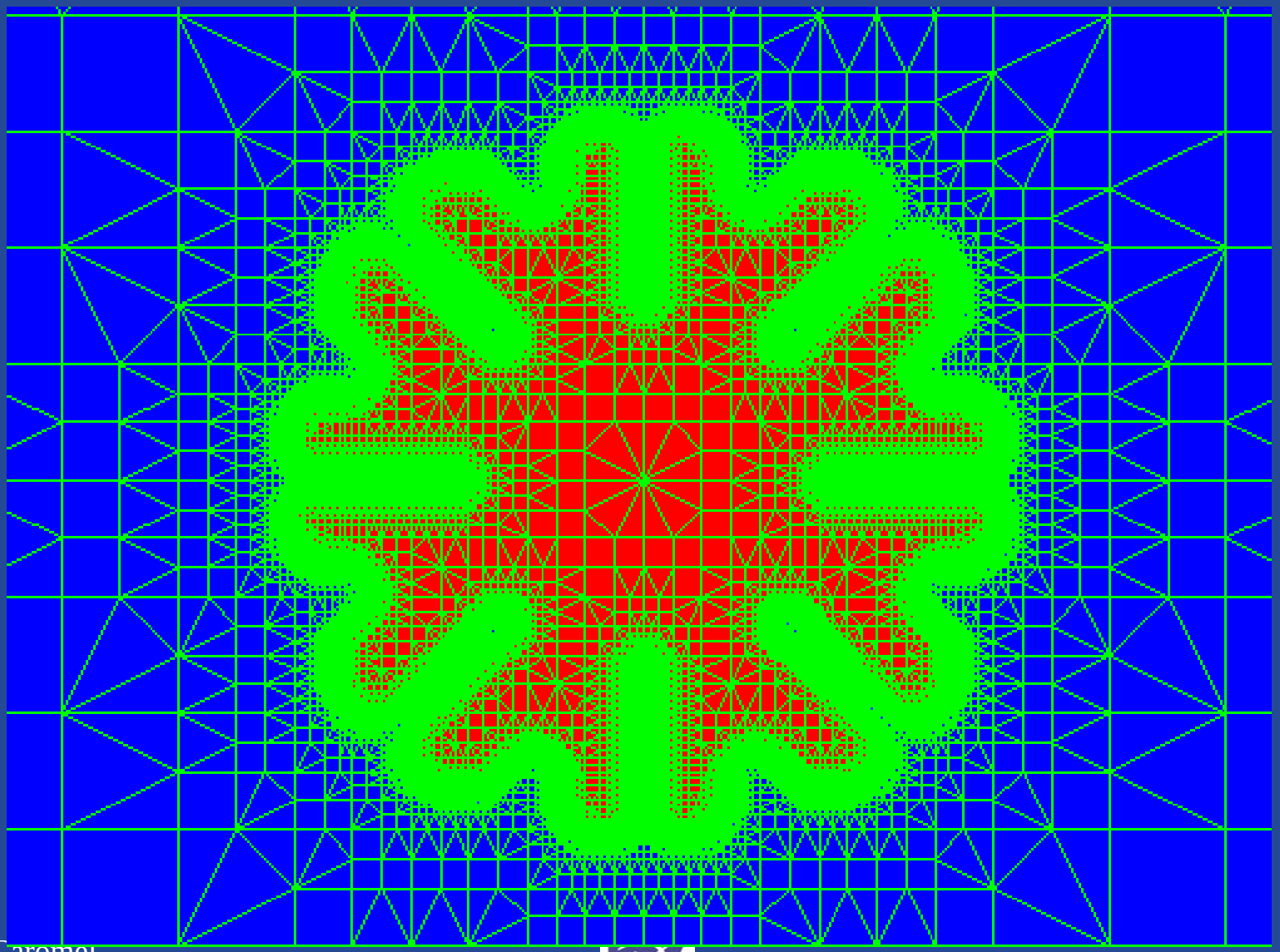
So, if your Middleware is adaptive, it can indeed be

Maladaptive!

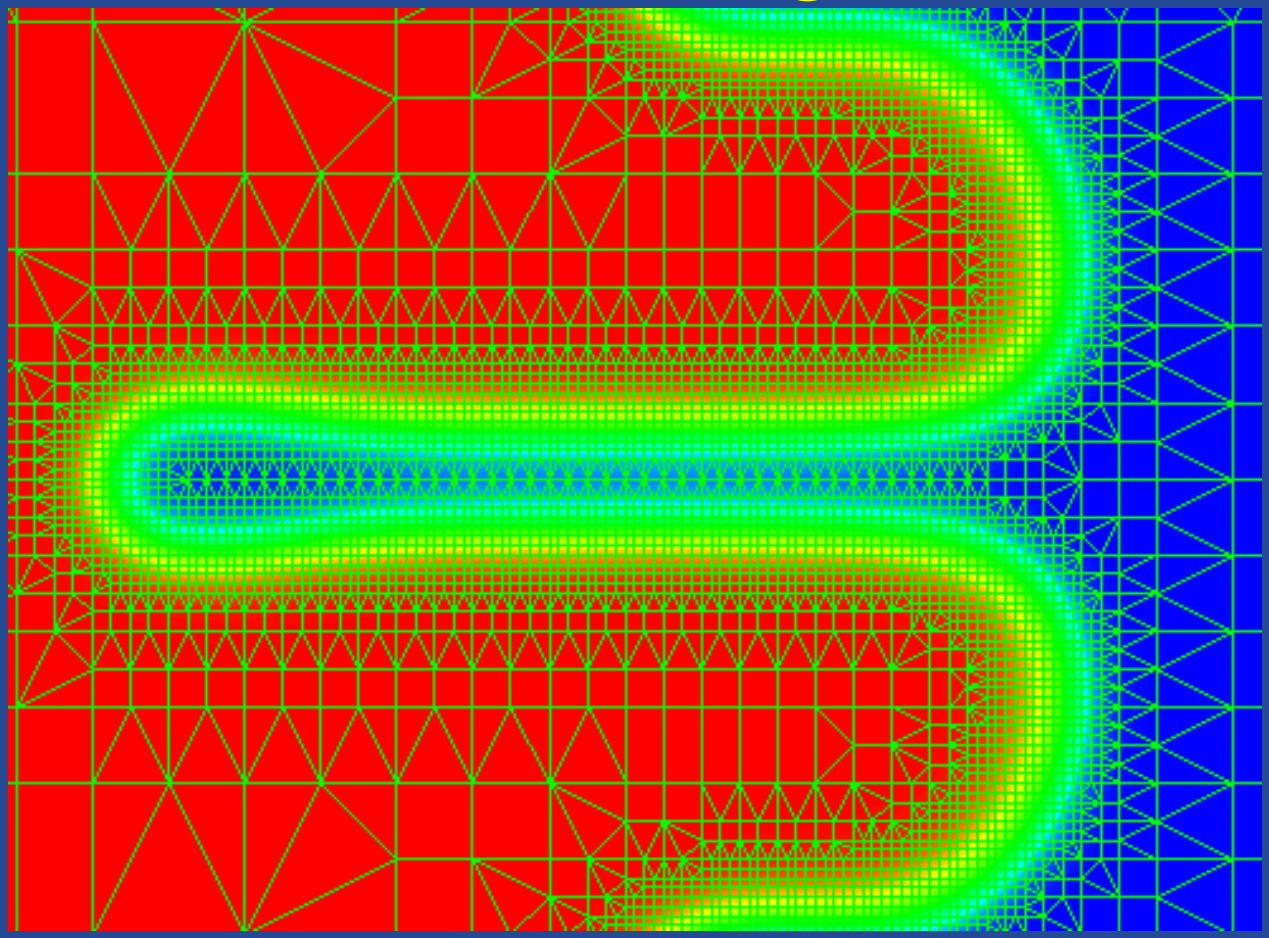
Adaptive Grids

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.:

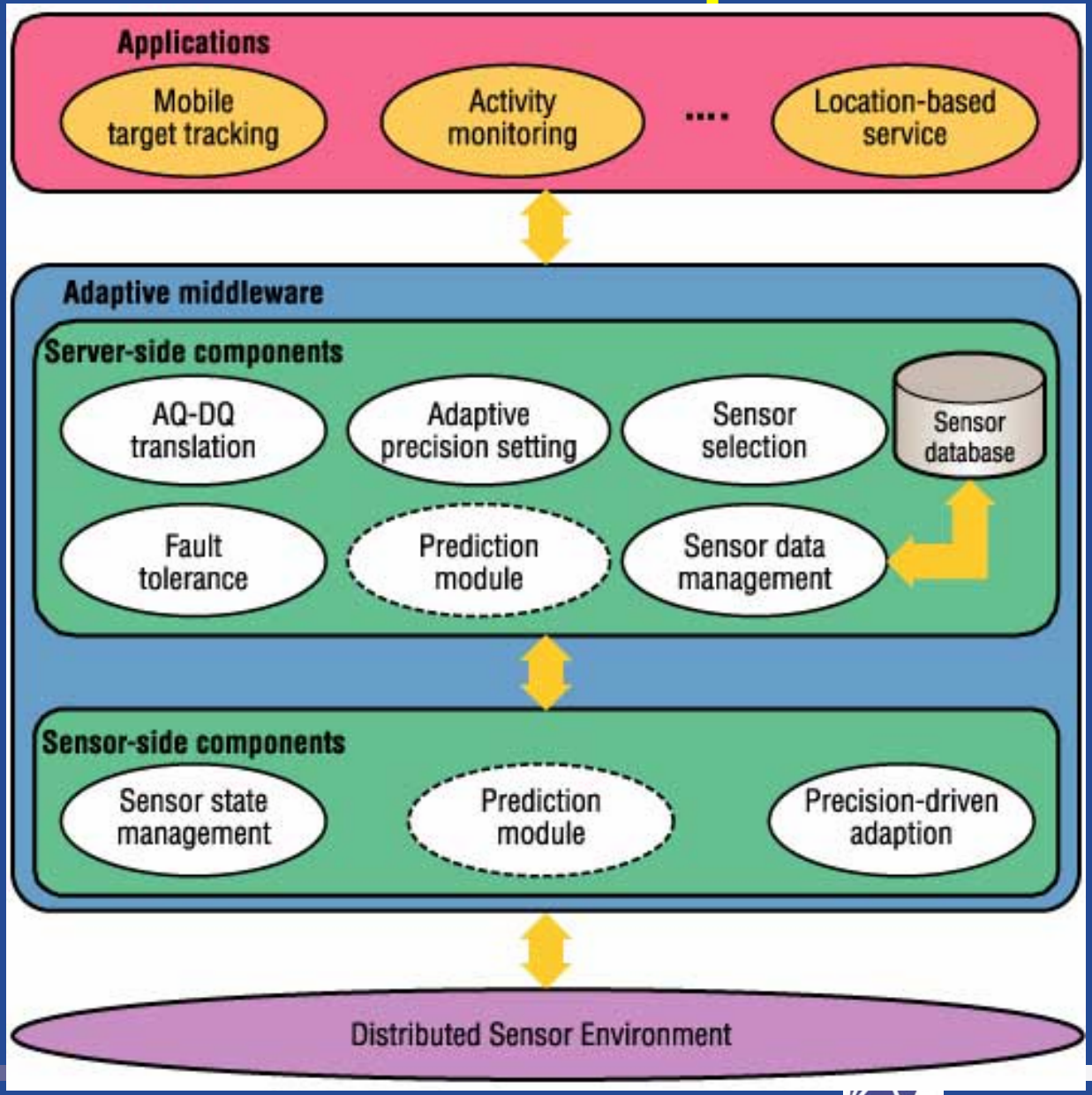
- Adaptive Numerical Method ==> Mesh Refinement
- Adaptive Multi-scale Bio-simulation
- Adaptive Discretisation Schemes
- …

How to make sure

Adaptive Applications do not confuse Adaptive Middelwares ?

ending up into … Maladaptive Grid !

What ’’Adaptive Grids’’ are now



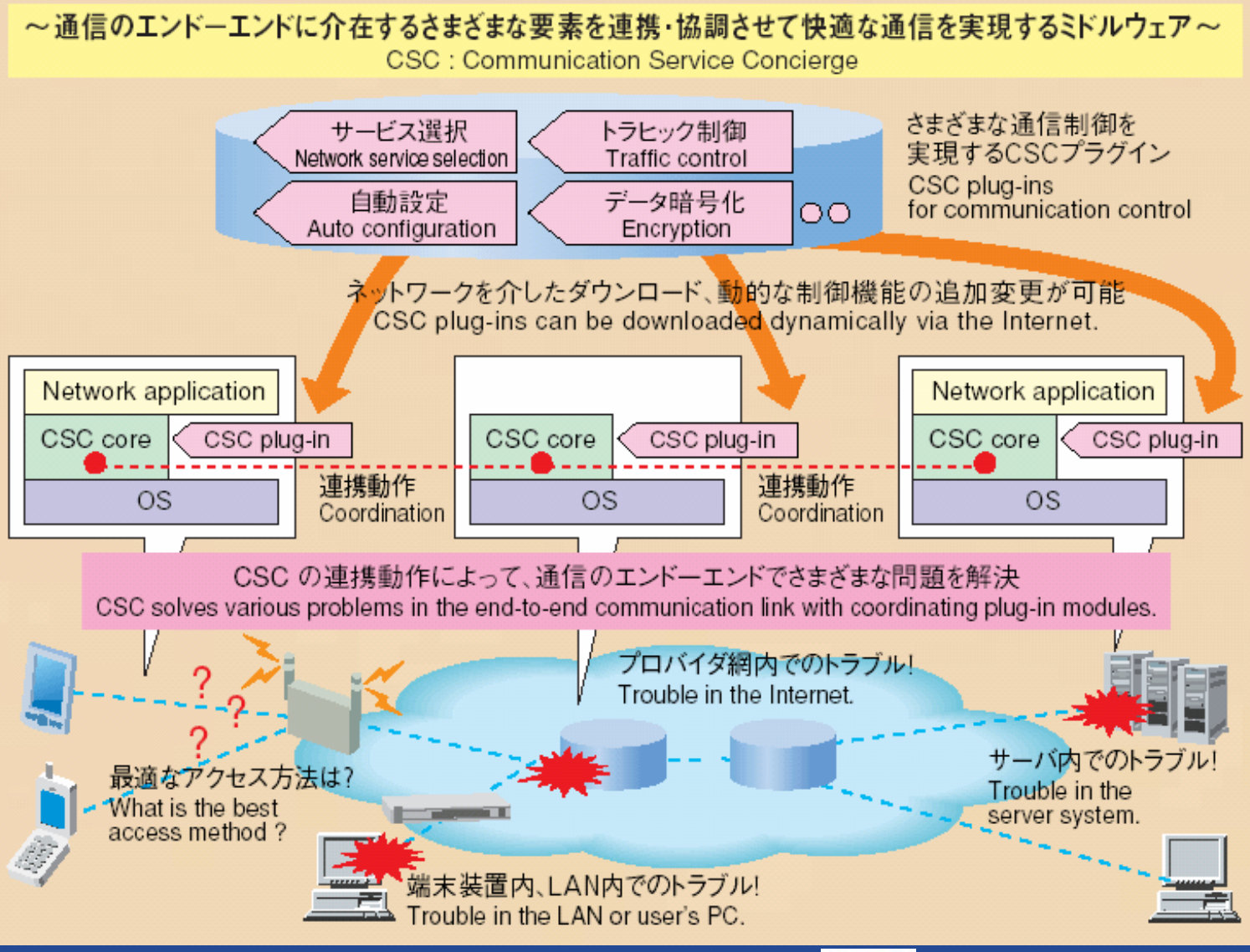
Adaptive Middleware for Distributed Sensor Environments

Xingbo Yu, Koushik Niyogi, Sharad Mehrotra, Nalini Venkatasubramanian

University of California, Irvine

IEEE Distributed System Online May 2003

”Adaptive Grids” are complex systems



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

W r a p p i n g

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:

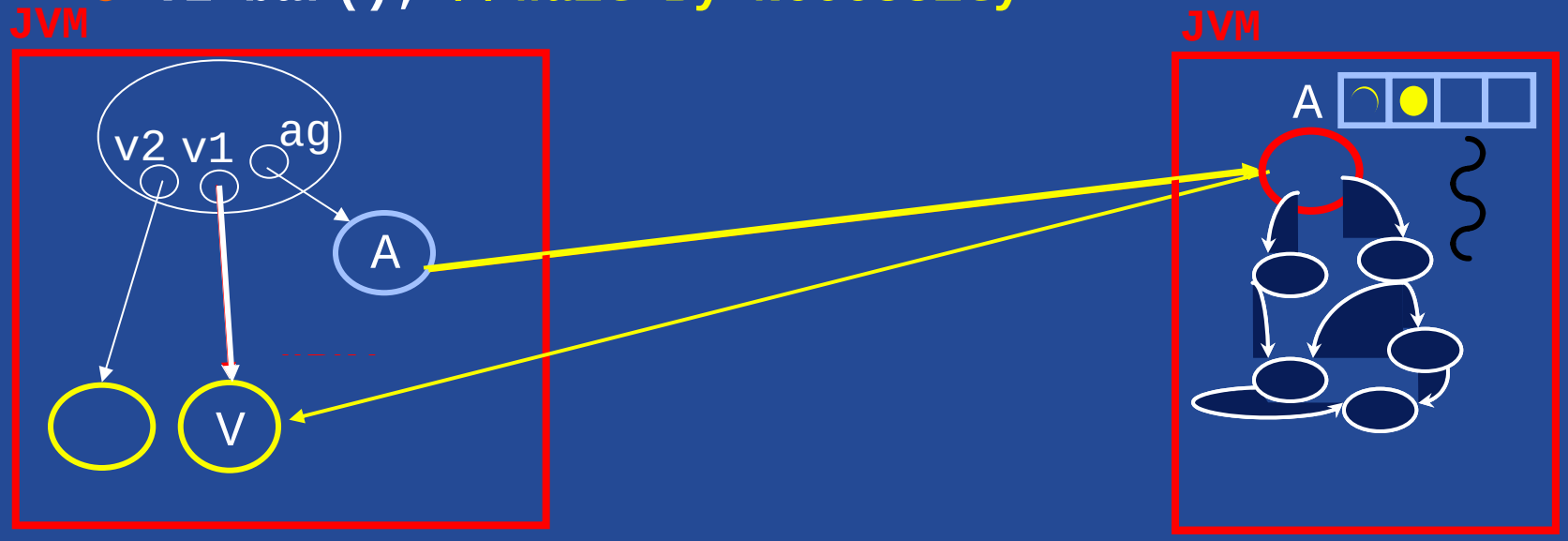
  Implicit Futures as method results
- Wait-By-Necessity:
  - Automatic wait upon a strict operation on an unknown future
  - First-Class Futures:

    - 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
```



Java Object

Active Object

Req. Queue

Future Object

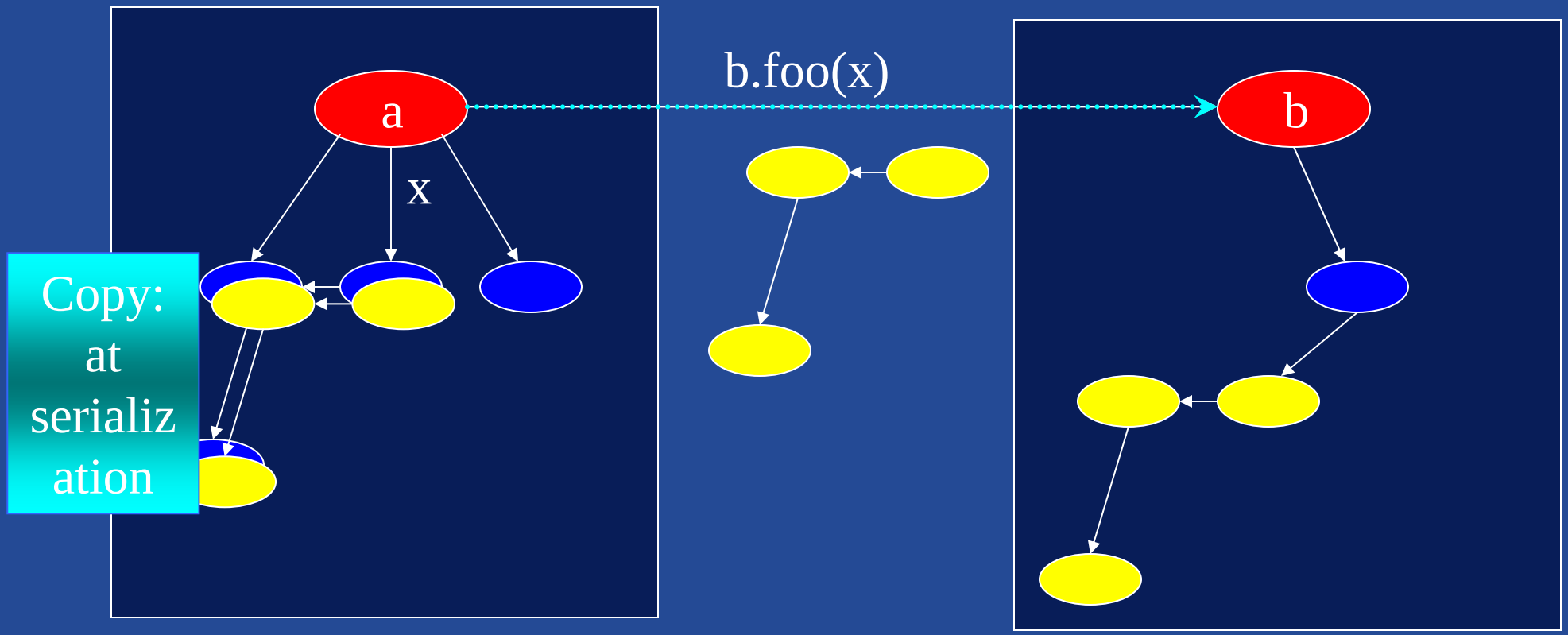
Proxy

Request

Thread

Wait-By-Necessity is a Dataflow Synchronization

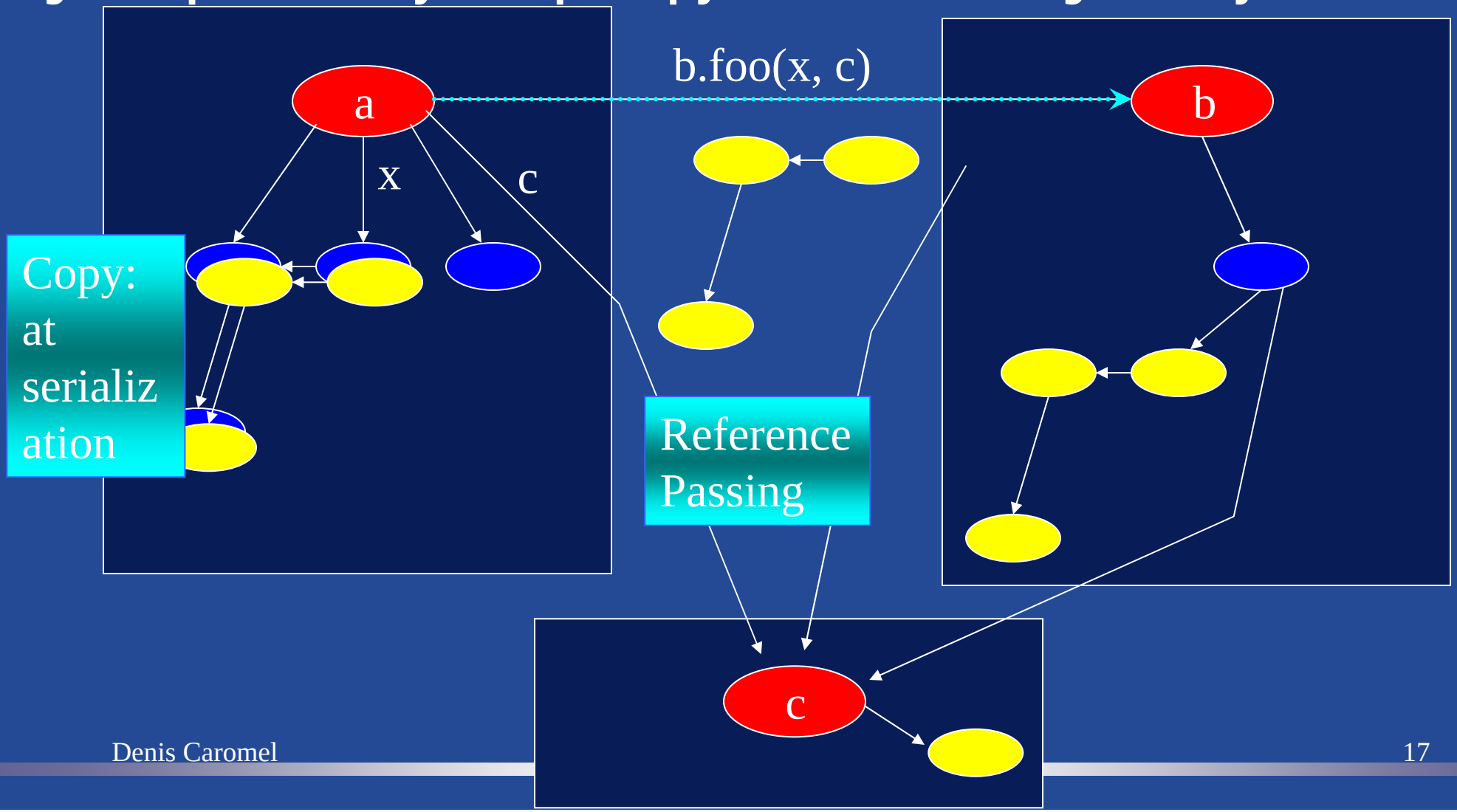
Call between Objects: Parameter passing: Copy of Java Objects



(Deep) Copies evolve independently -- No consistency

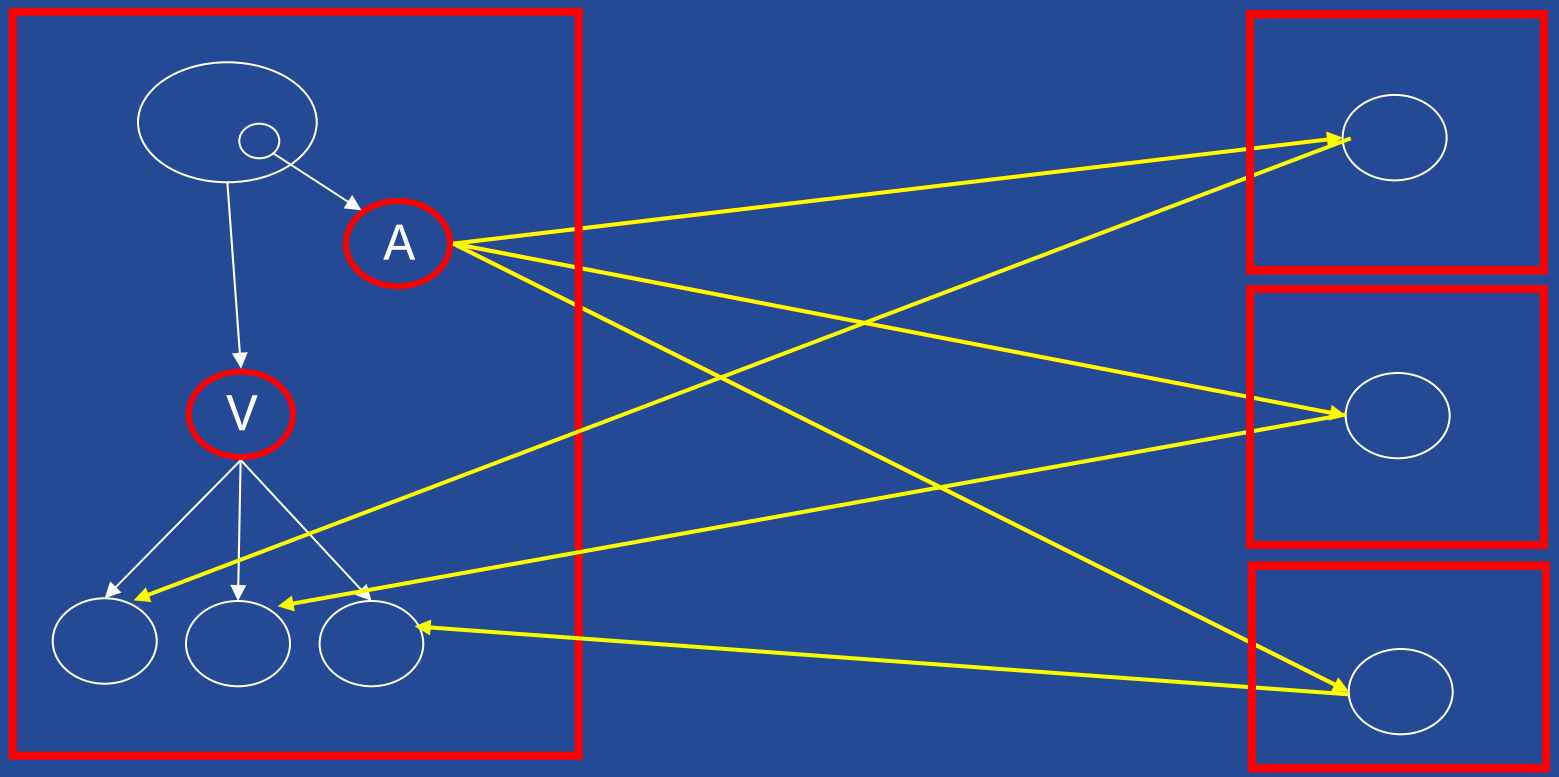
Call between Objects: Parameter Passing: Active Objects

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
```



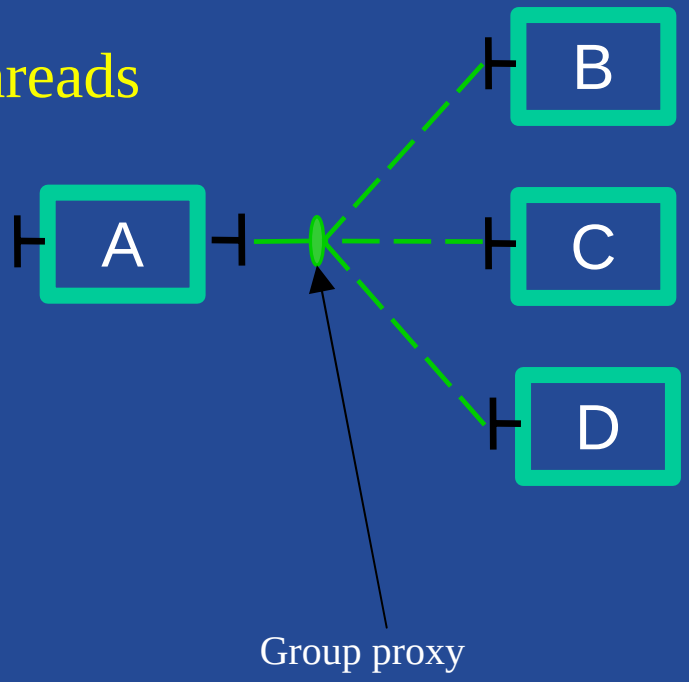
Object-Oriented
Typed Group Communications

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

Adaptive Feature 1: Parallel Group Communications

Adaptive strategy to manage the number of threads

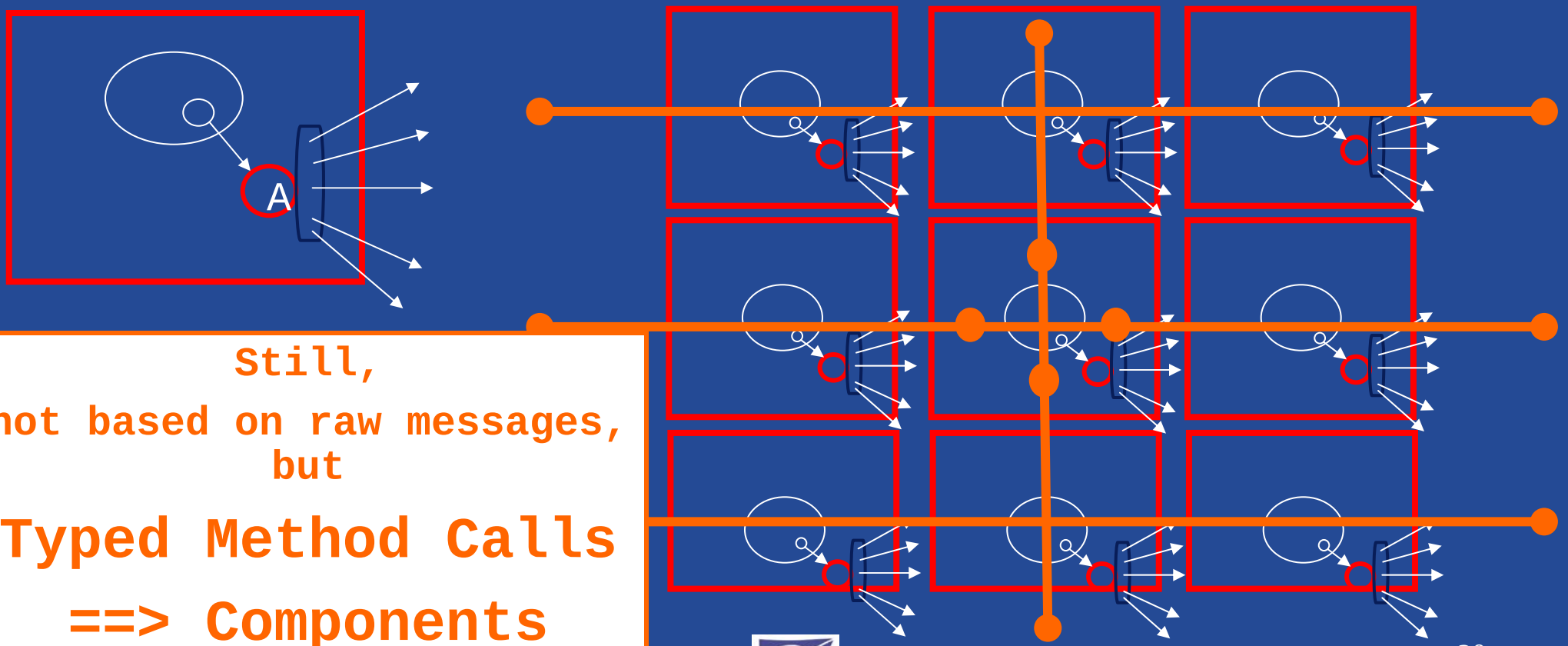
Execution of N calls in //



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");
```



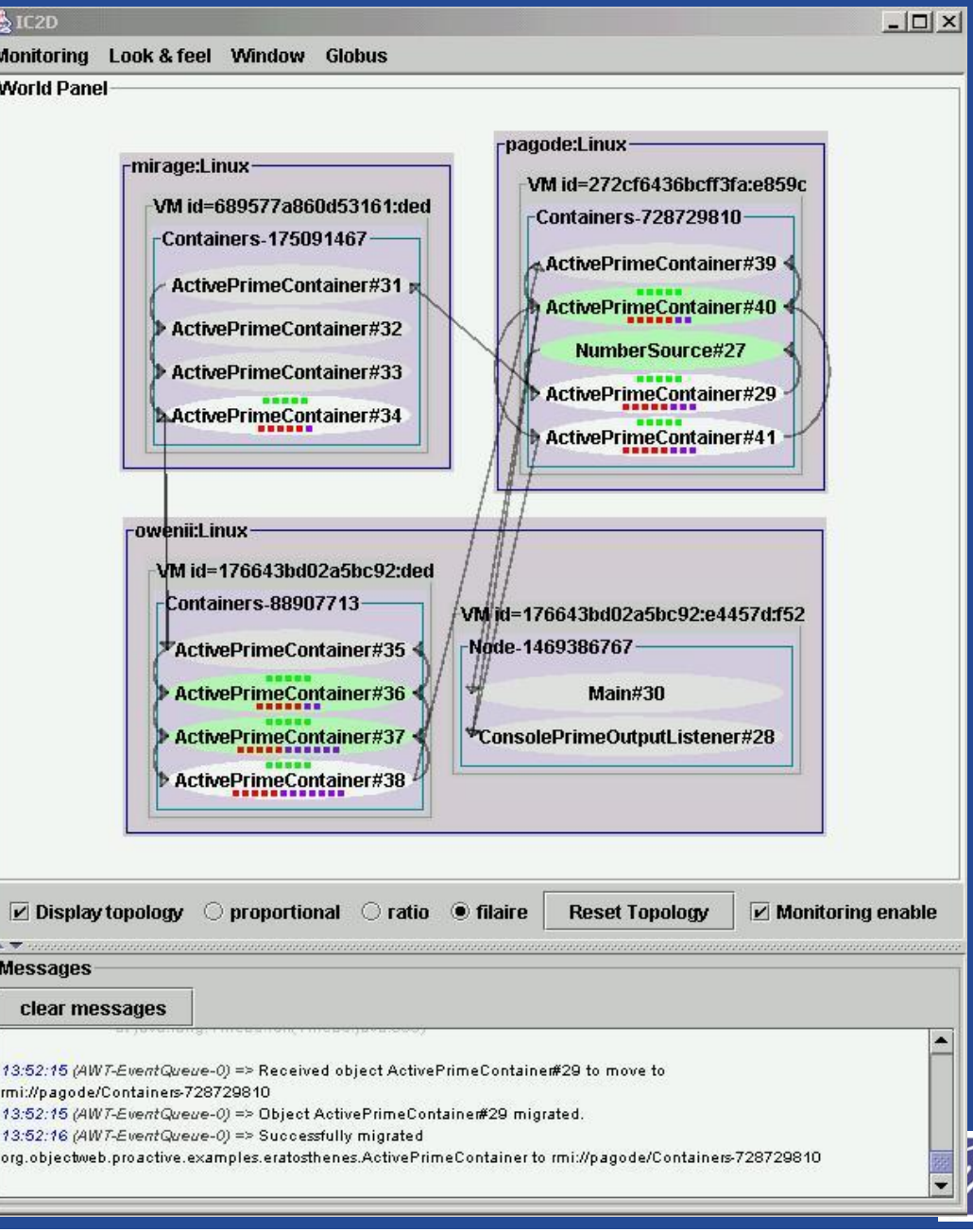
Still,

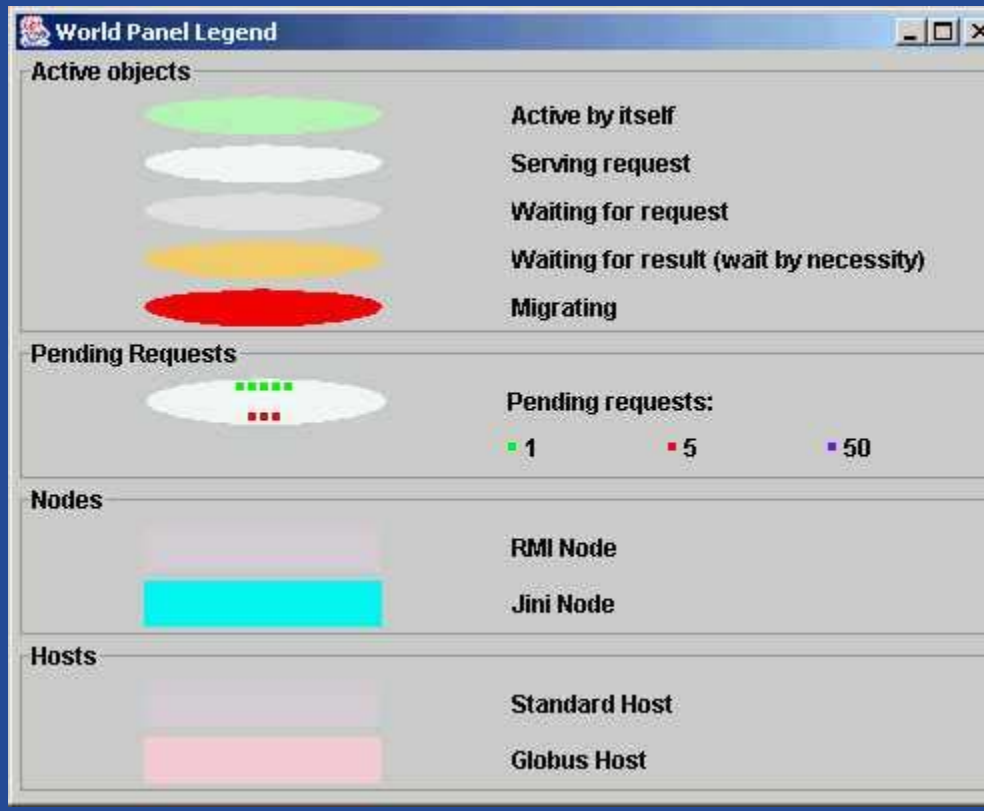
not based on raw messages, but

Typed Method Calls

==> Components

IC2D: Interactive Control and Debugging of Distribution





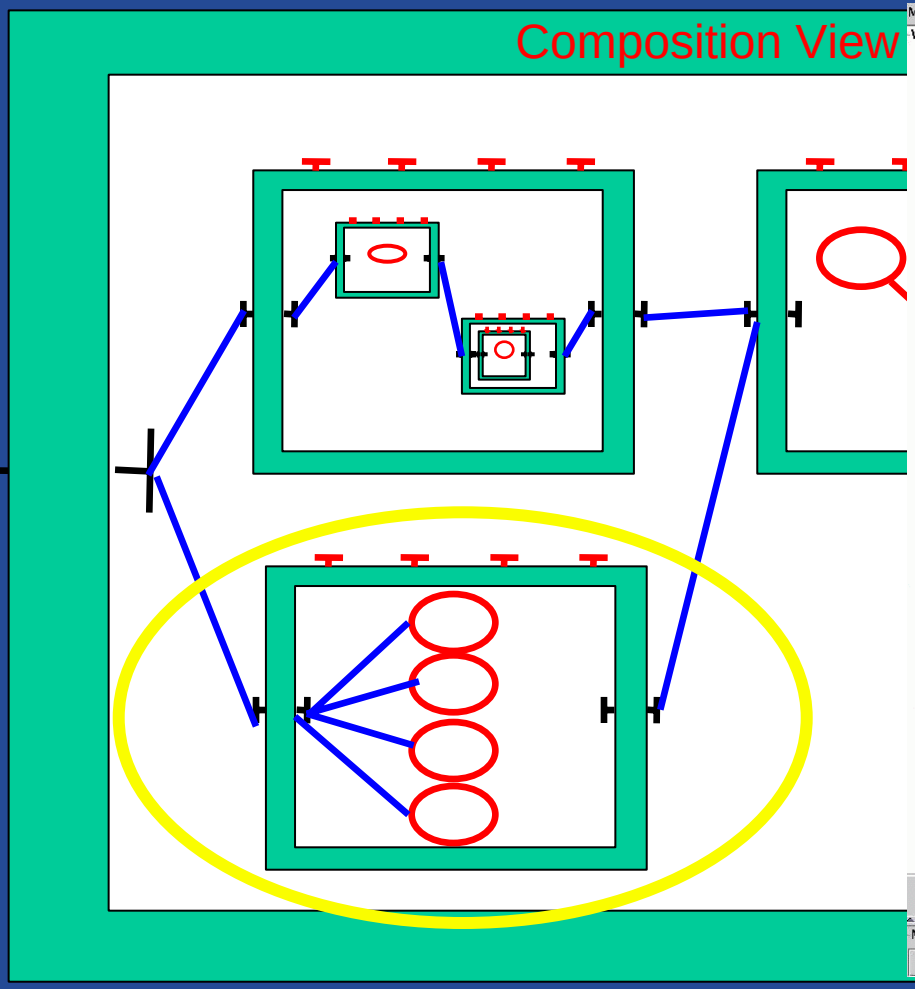
With any ProActive application

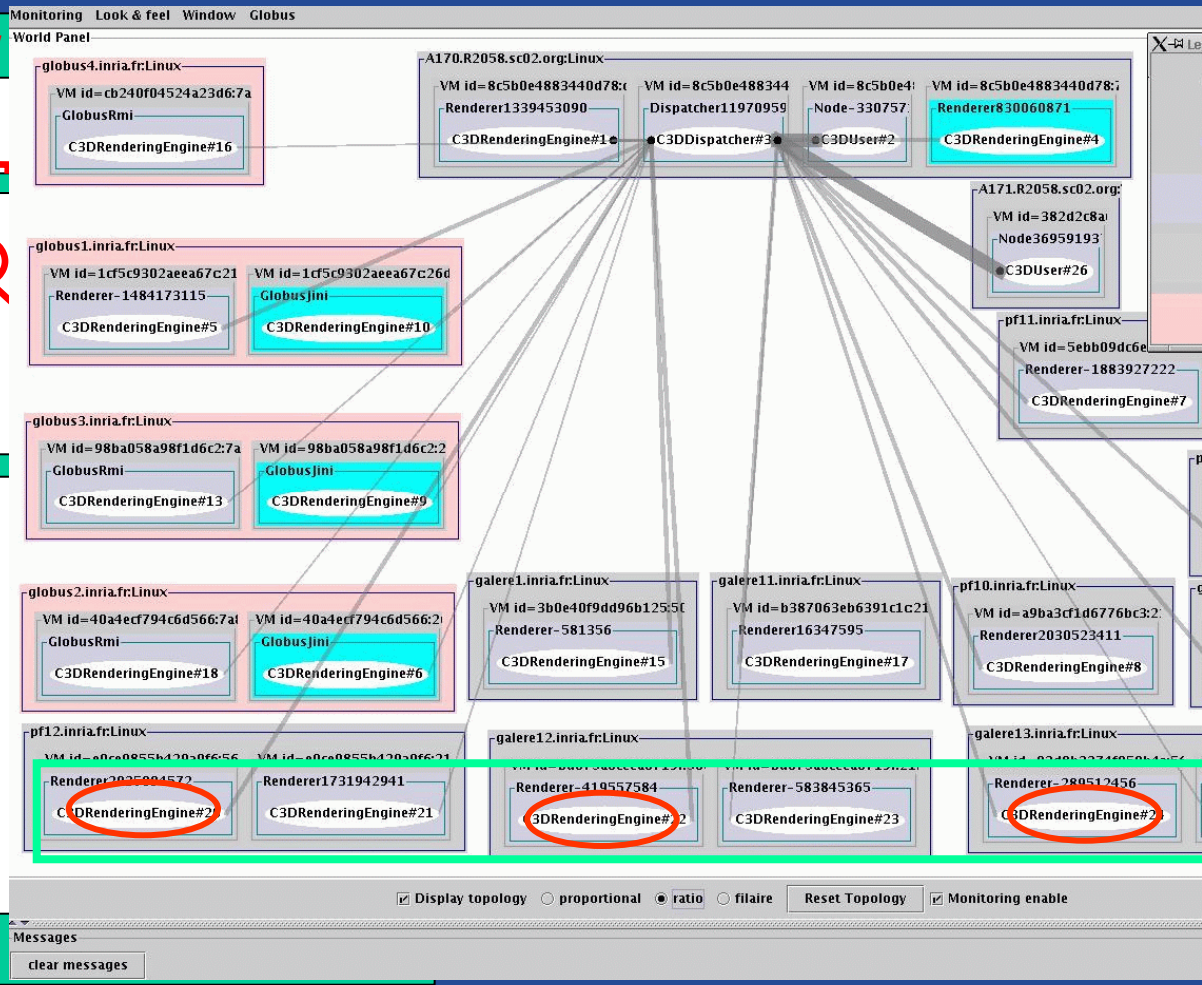
Features:

Graphical and Textual visualization

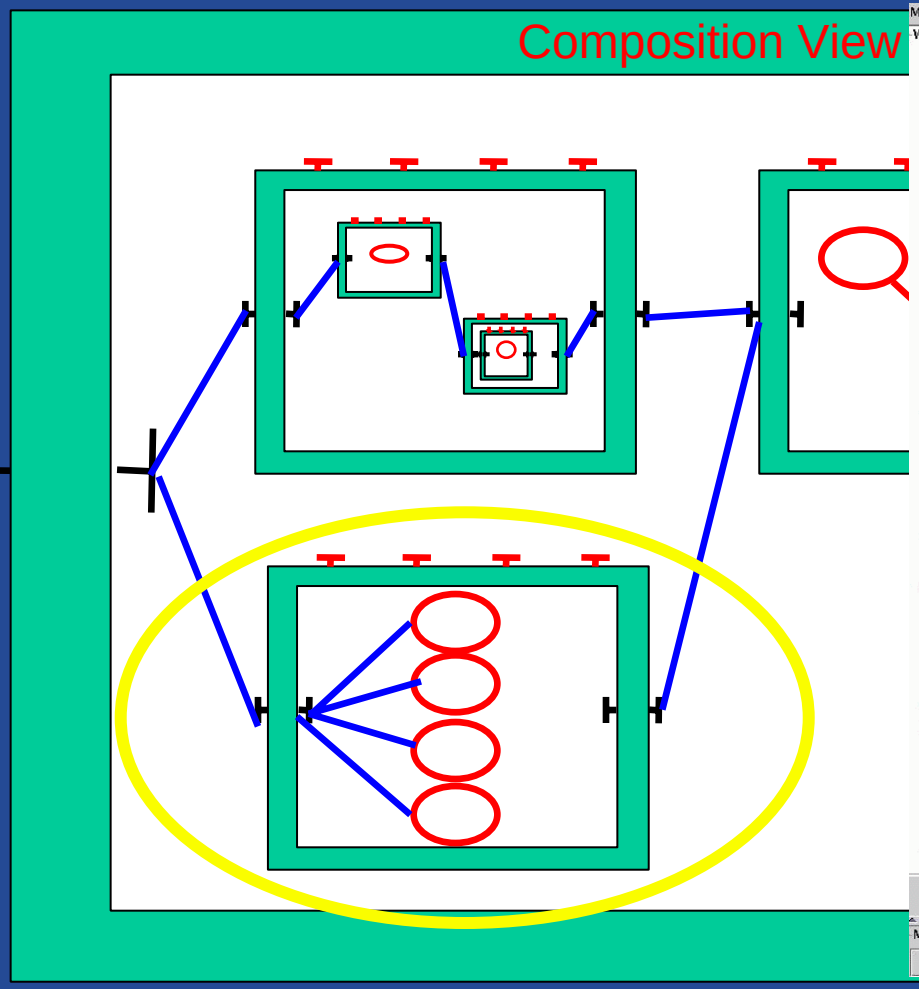
Monitoring and Control

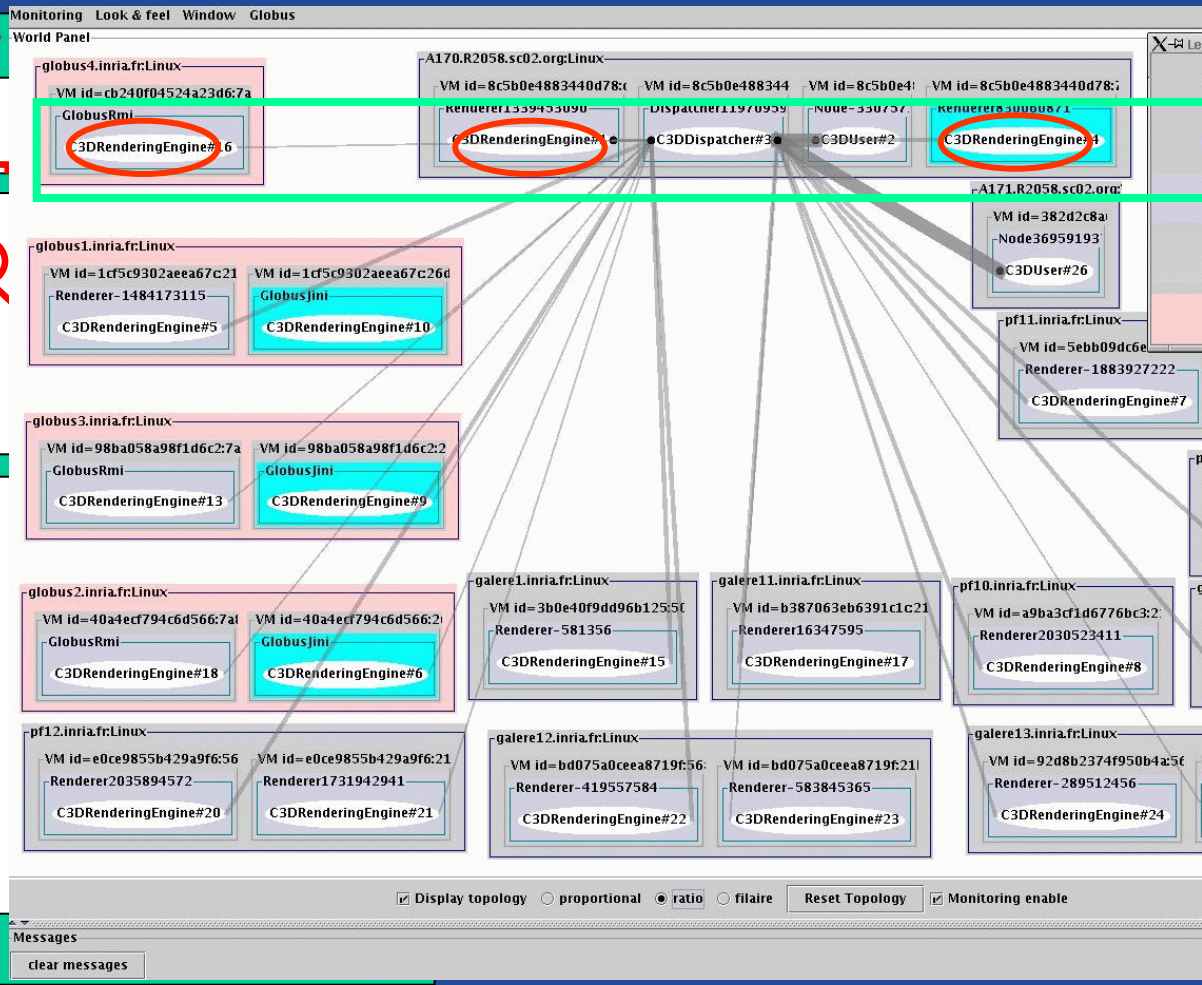
Distributed Components
Graphical Composition, Monitoring, Migration





Distributed Components
Graphical Composition, Monitoring, Migration





Adaptive Feature 2: Multi-transports 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 fullfilled)

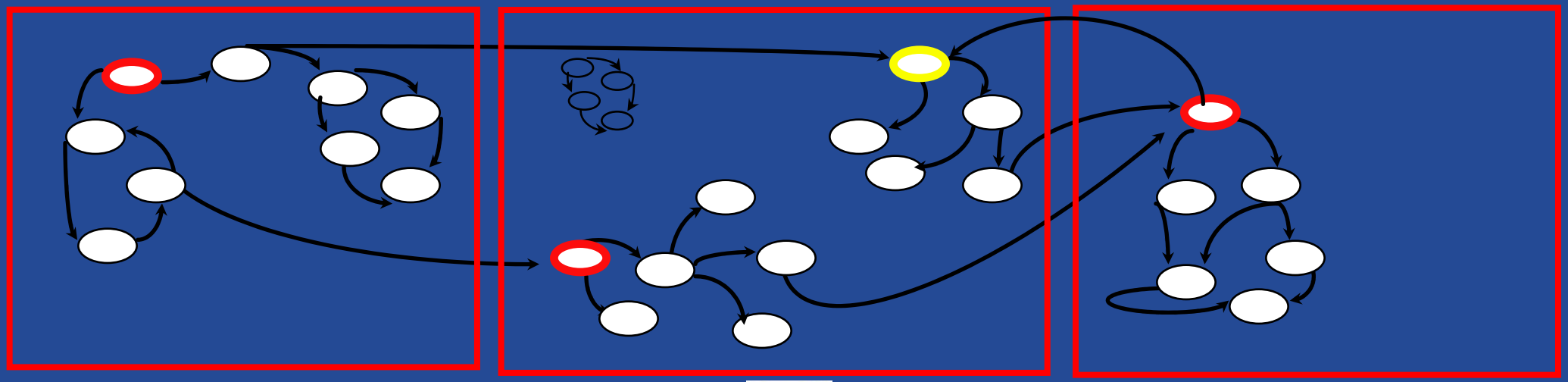
Principles and optimizations

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)



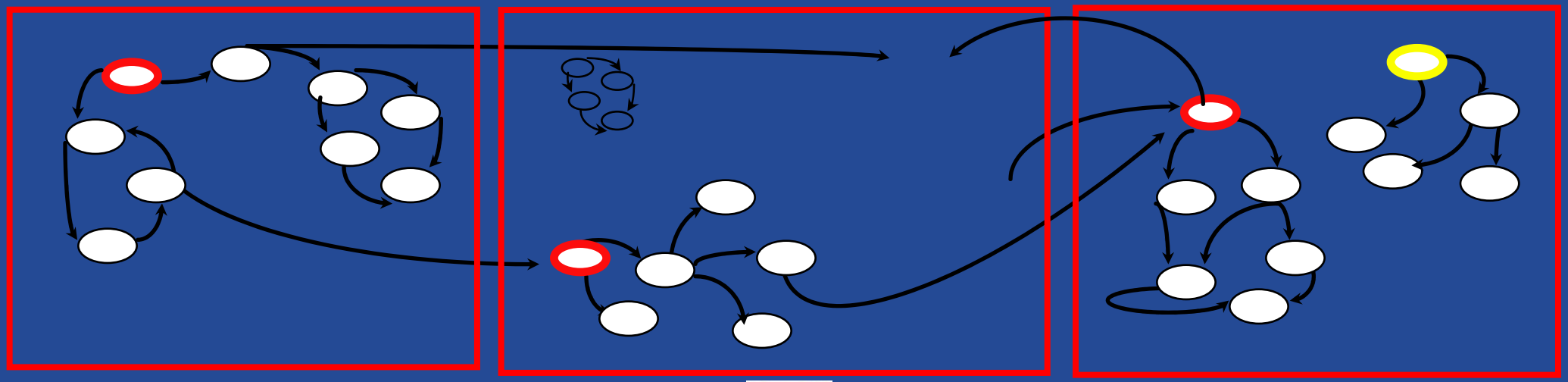
Principles and optimizations

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)



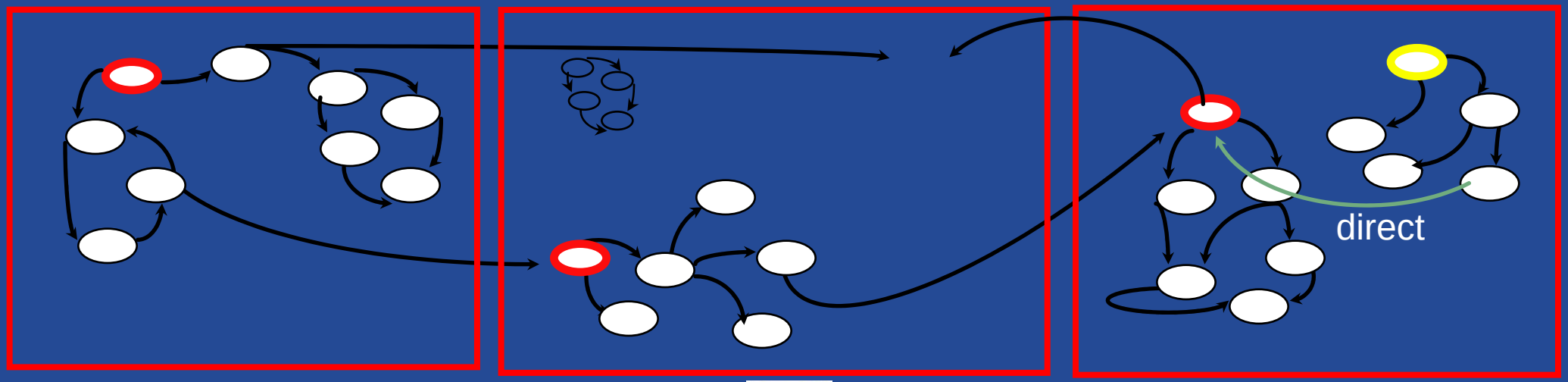
Principles and optimizations

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)



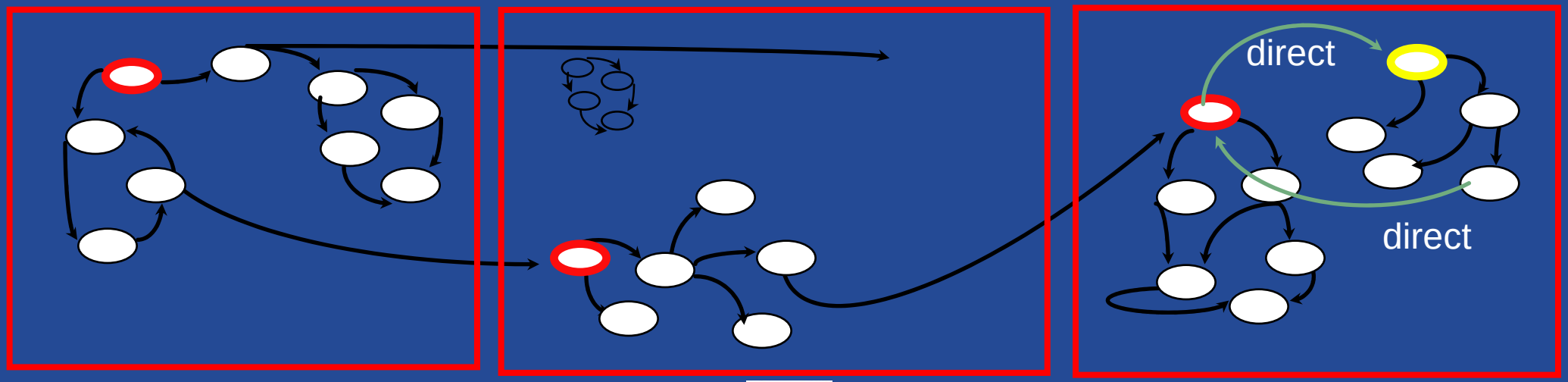
Principles and optimizations

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)



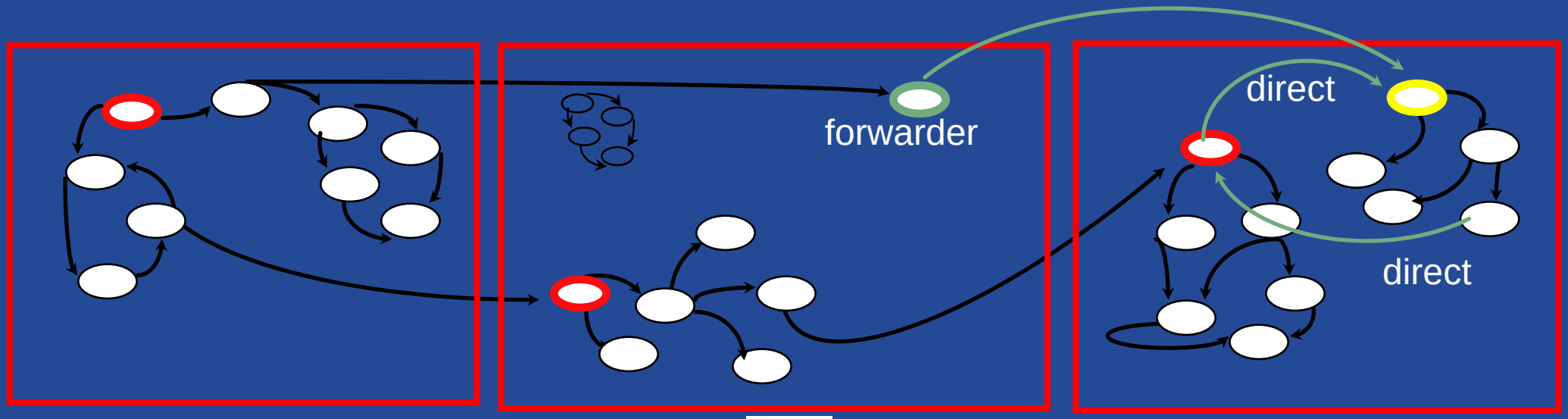
Principles and optimizations

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)



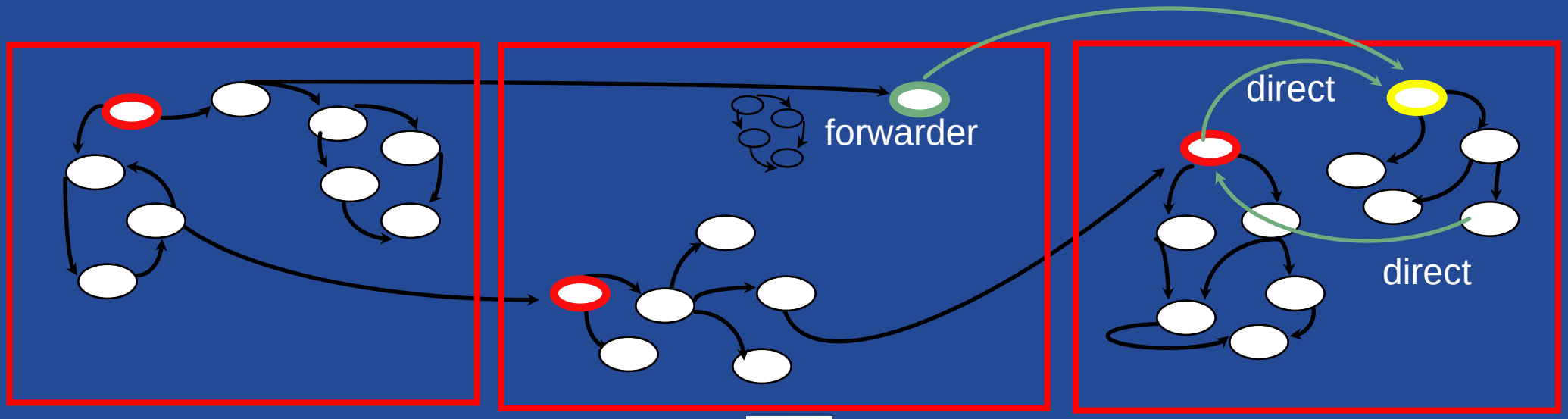
Principles and optimizations

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)



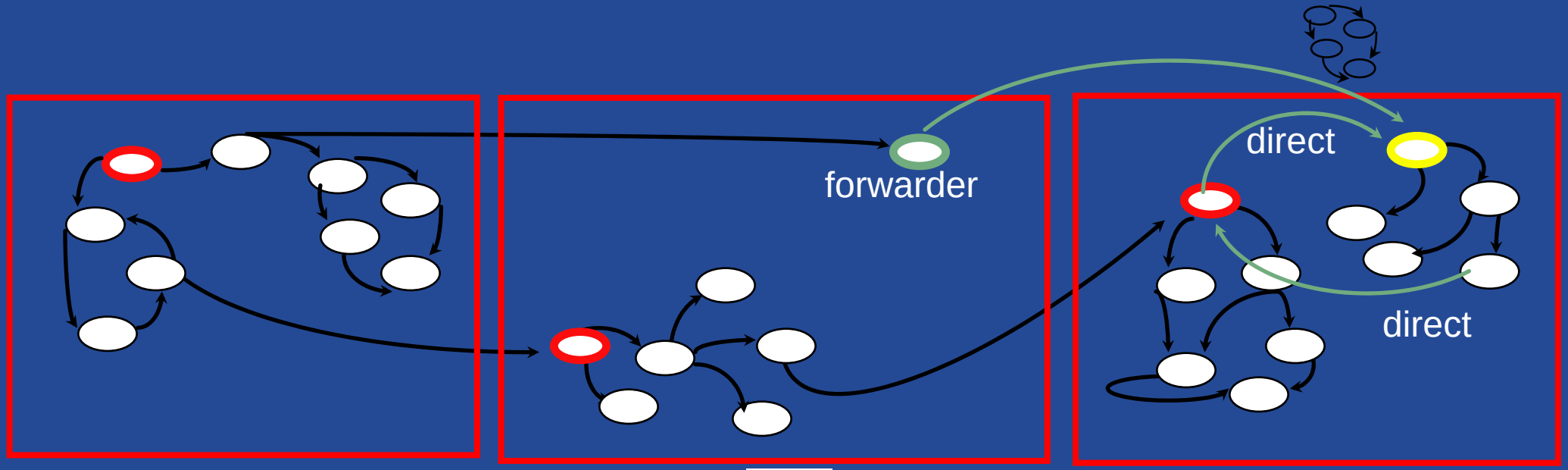
Principles and optimizations

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)



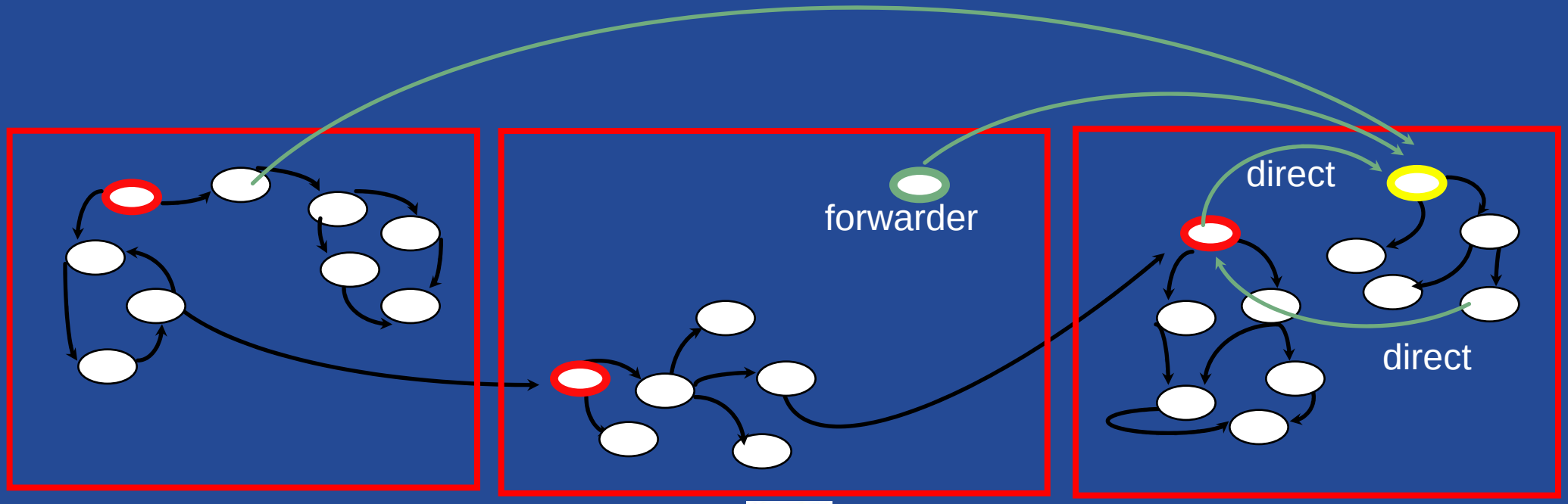
Principles and optimizations

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

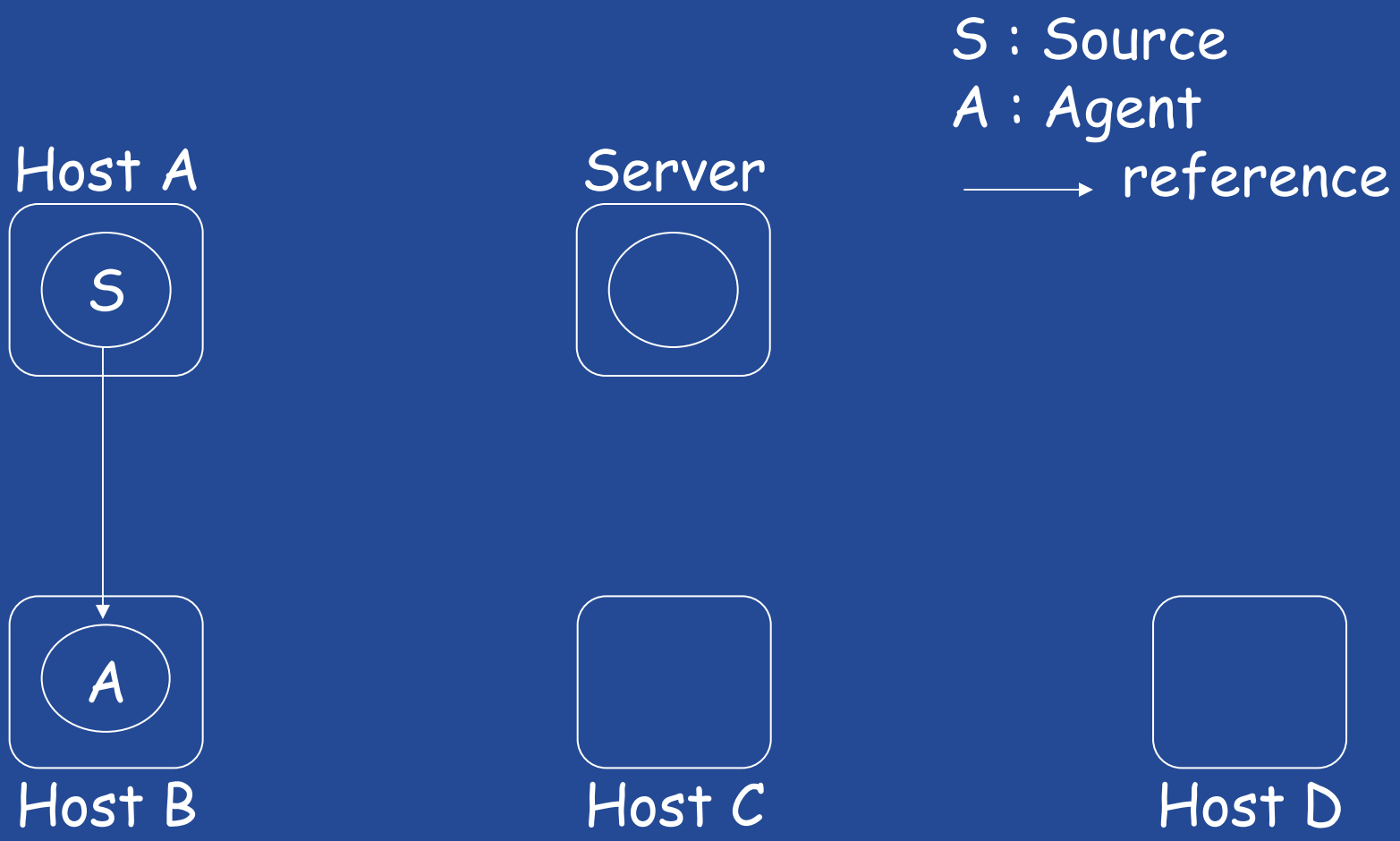
Safe migration (no agent in the air!)

Local references if possible when arriving within a VM

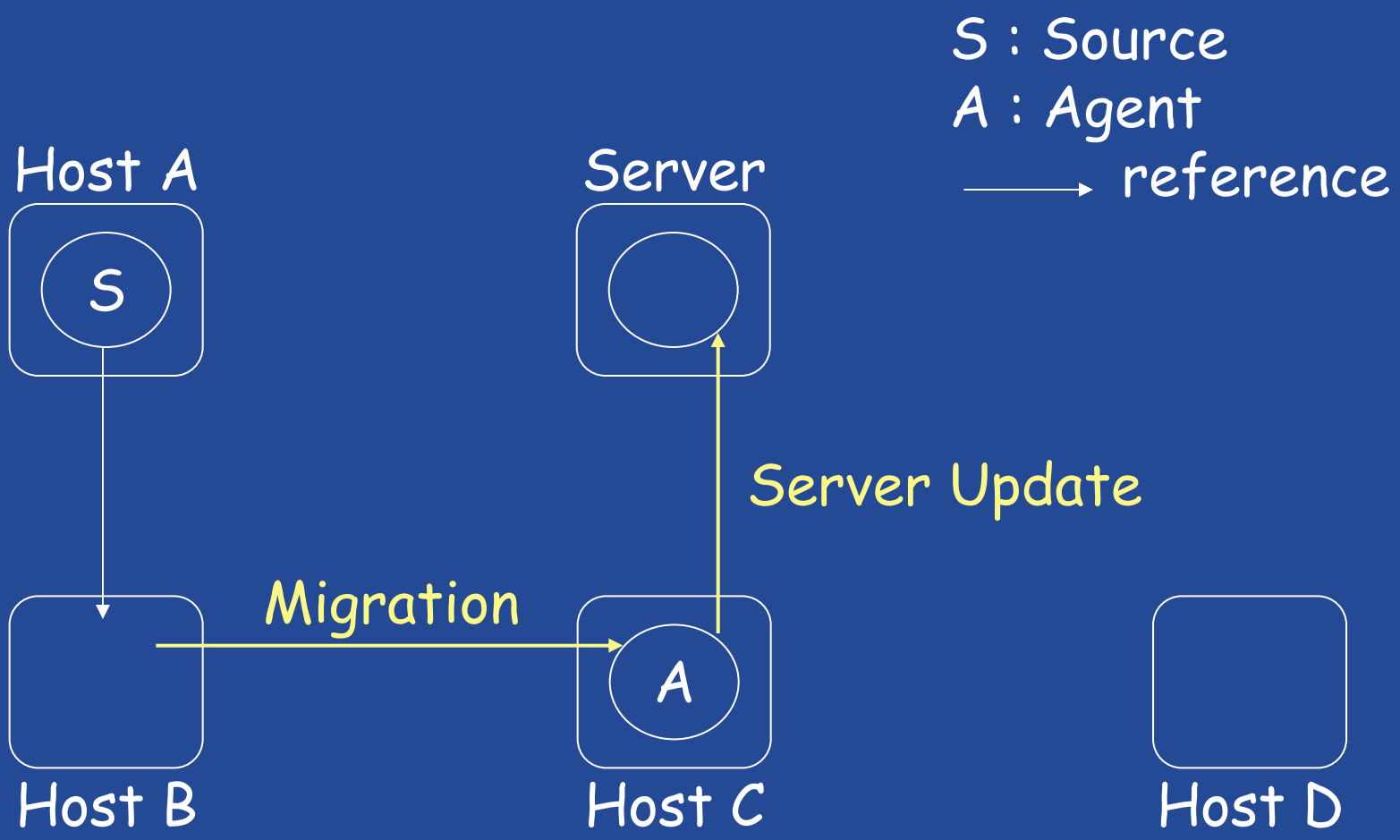
Tensionning (removal of forwarder)



Other Strategy: Centralized

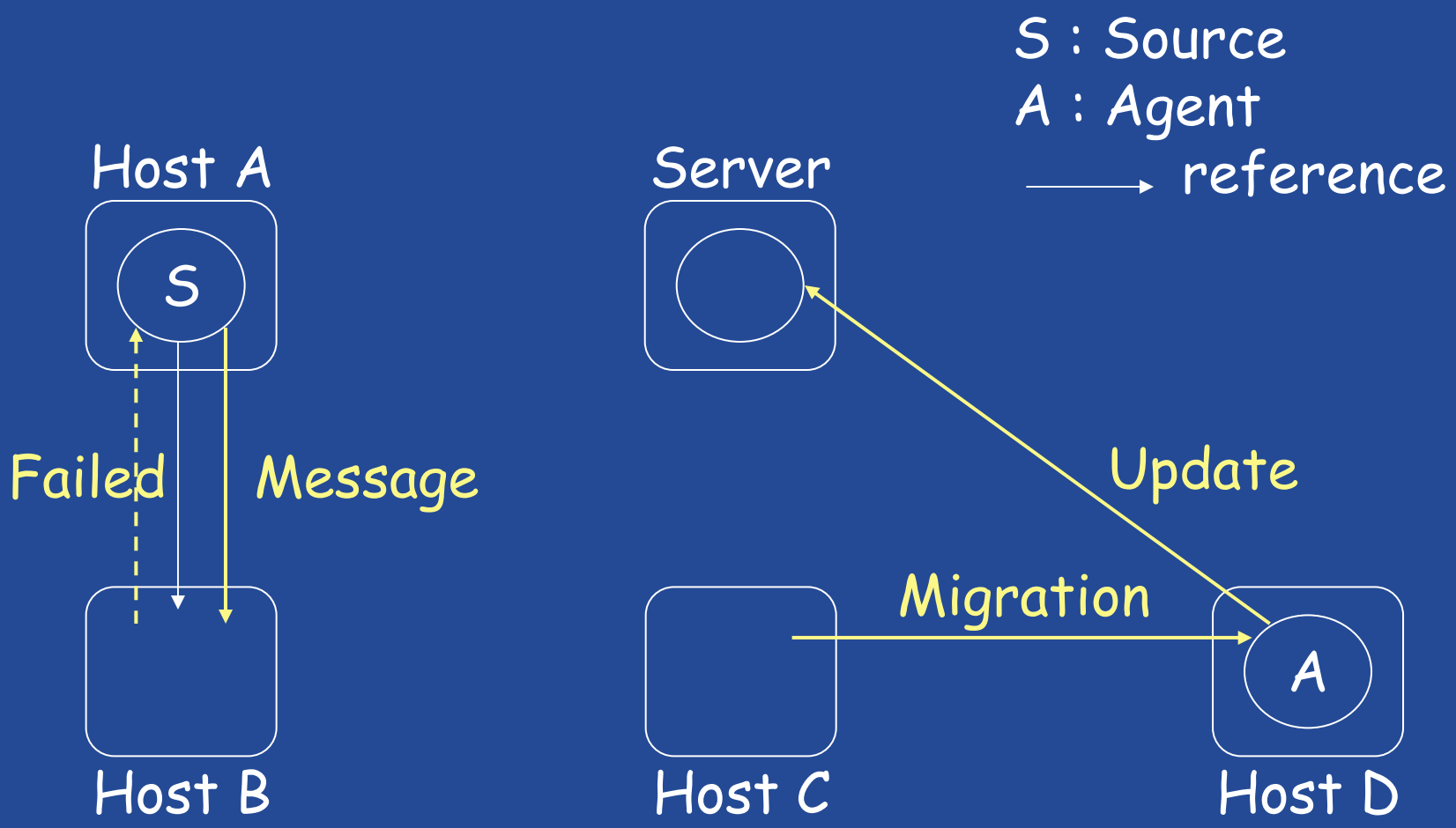


Centralized Strategy (2)



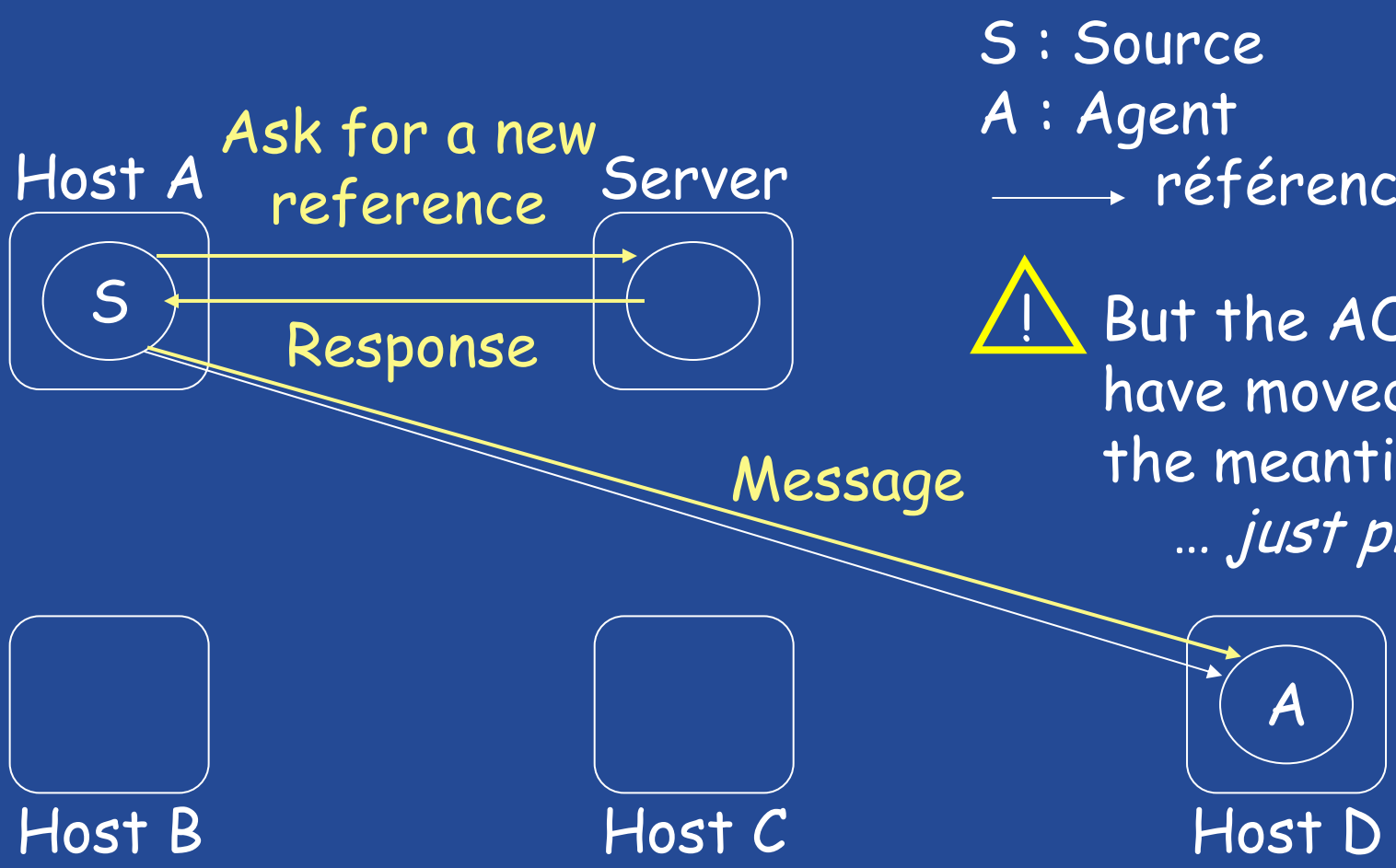
A migrating object updates the server

Centralized Strategy (3)



A migrating object updates the server

Centralized Strategy (4)



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

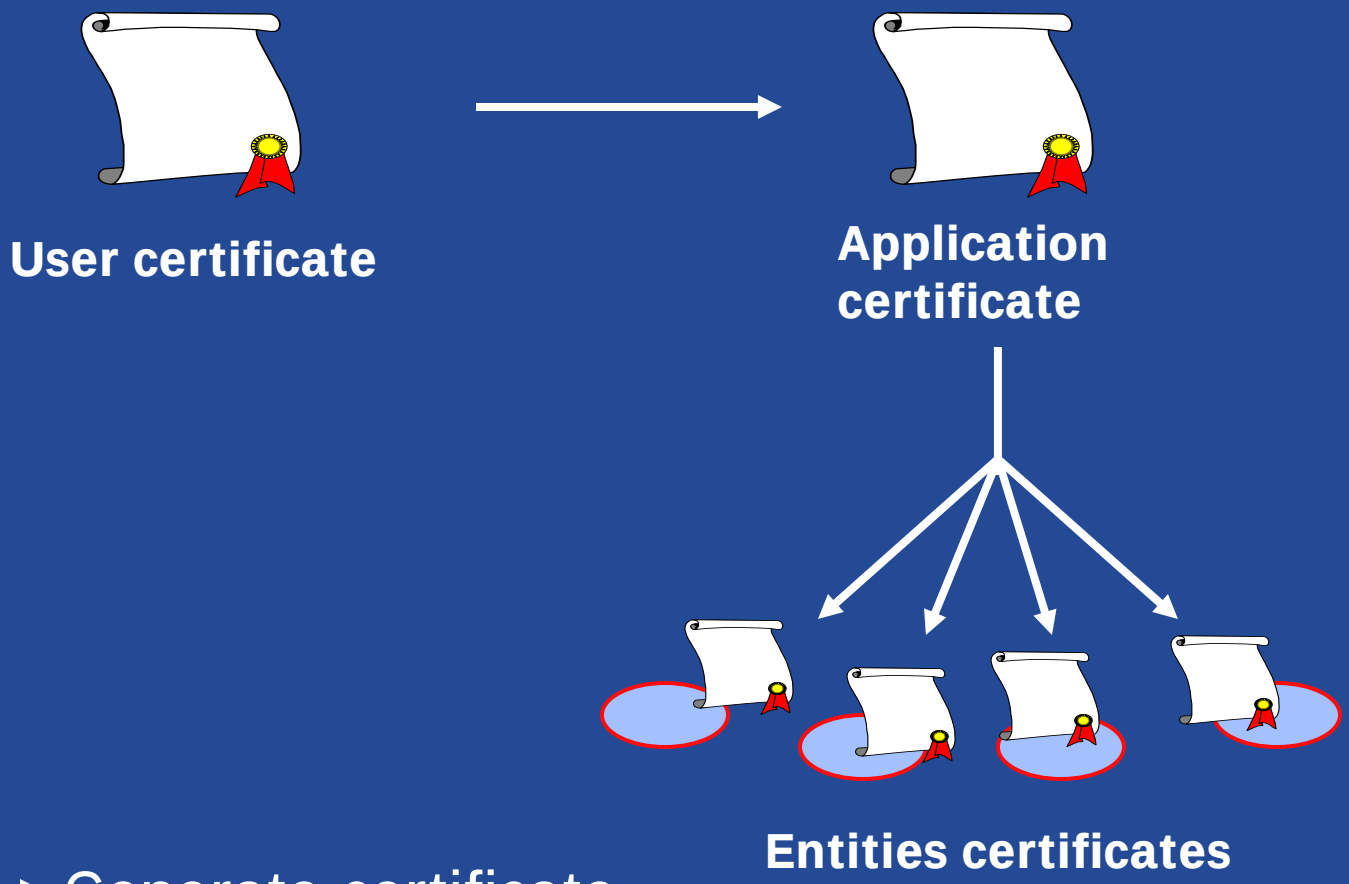
ProActive Security Features

- 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

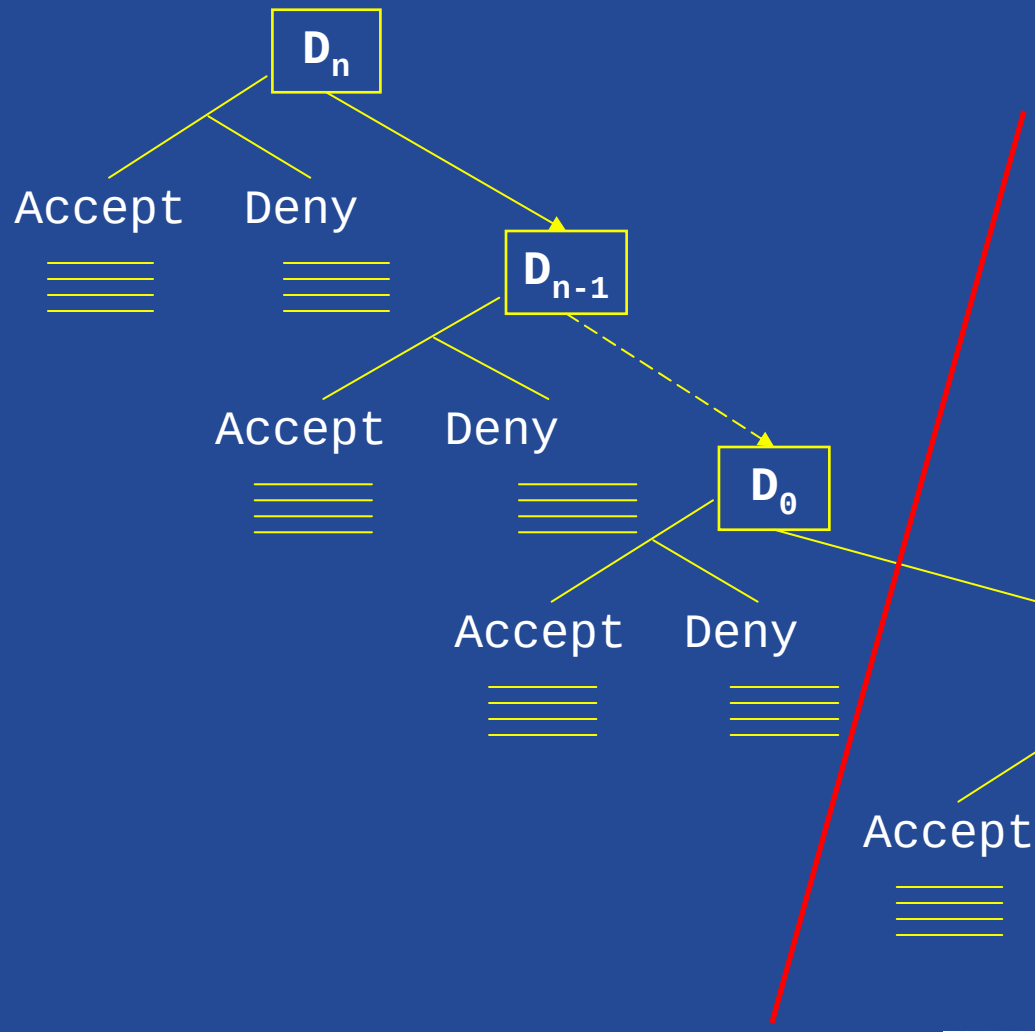
A Chain of X509 Certificates



→ Generate certificate

Multi-level Policies

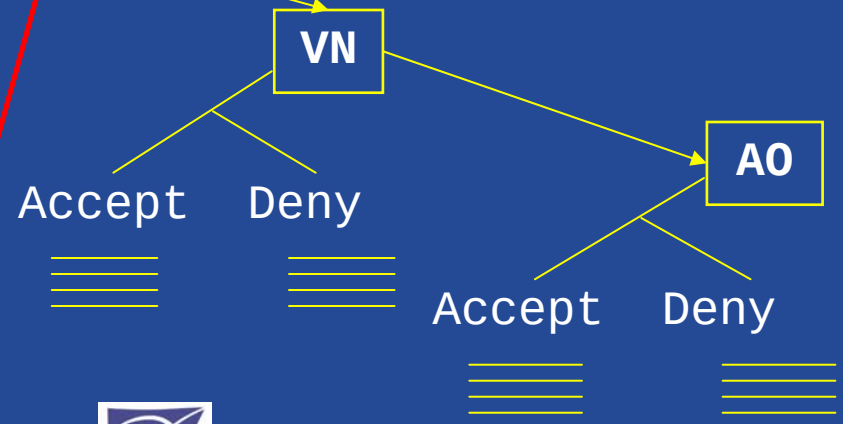
Administrator + User-level policy



Security policy is defined according all matching rules from:

- **Domains**
- **Virtual Nodes**
- **Active Objects**

Application-level policy



Security policy

Combining Policies

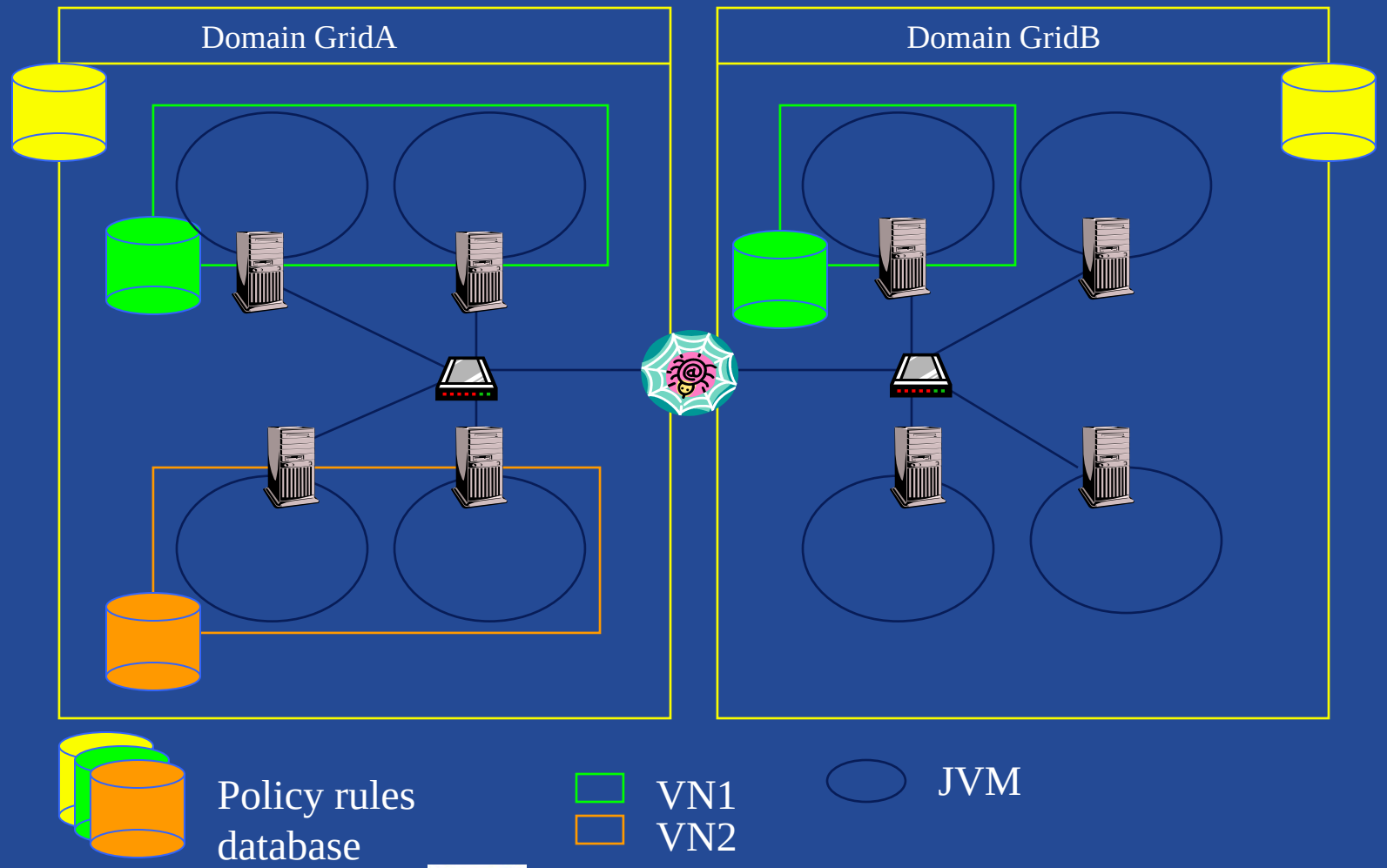
Search for the most specific rule in each domain.

Retrieve all matching rules

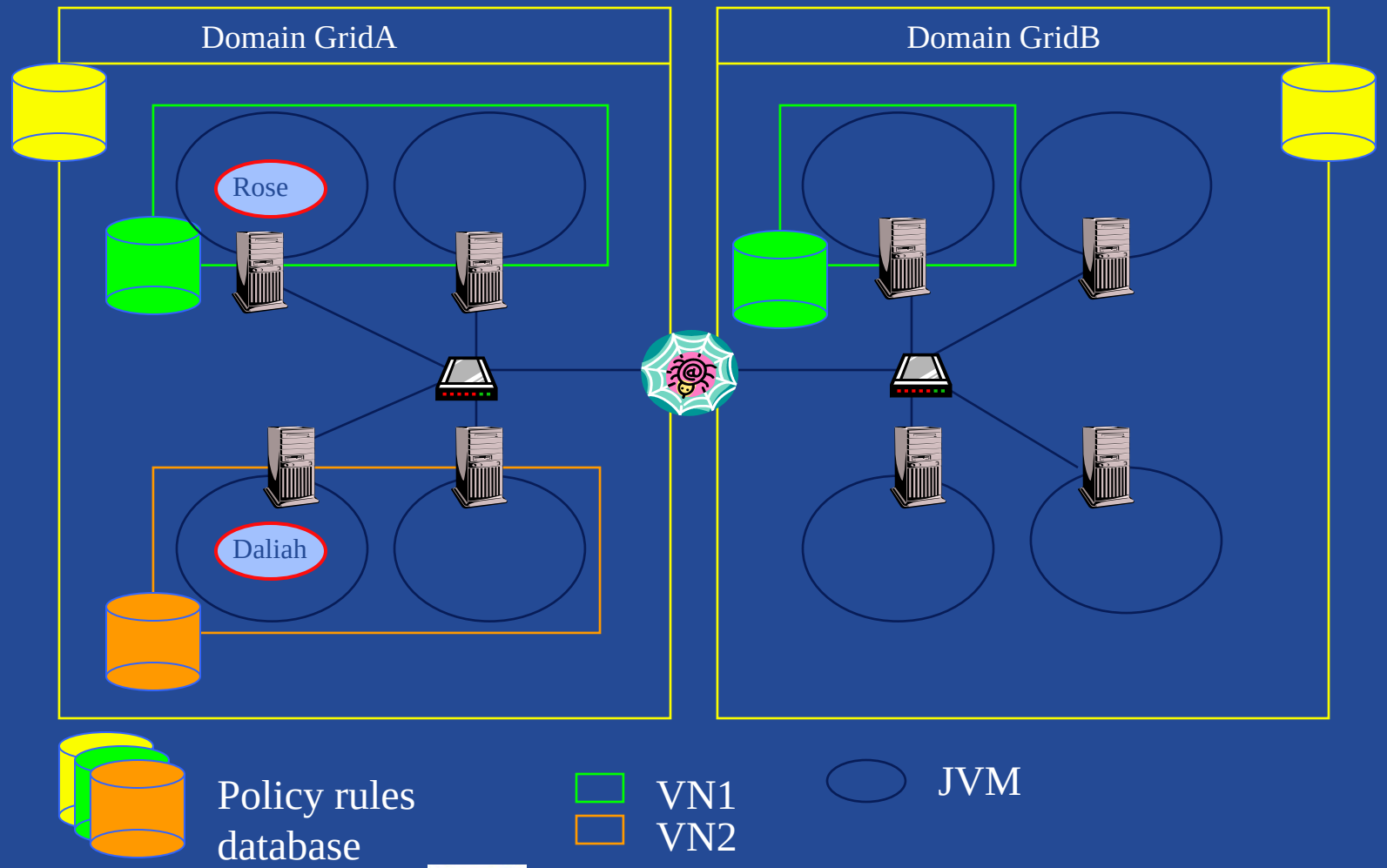
Compute policies according to security attributes

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

Example

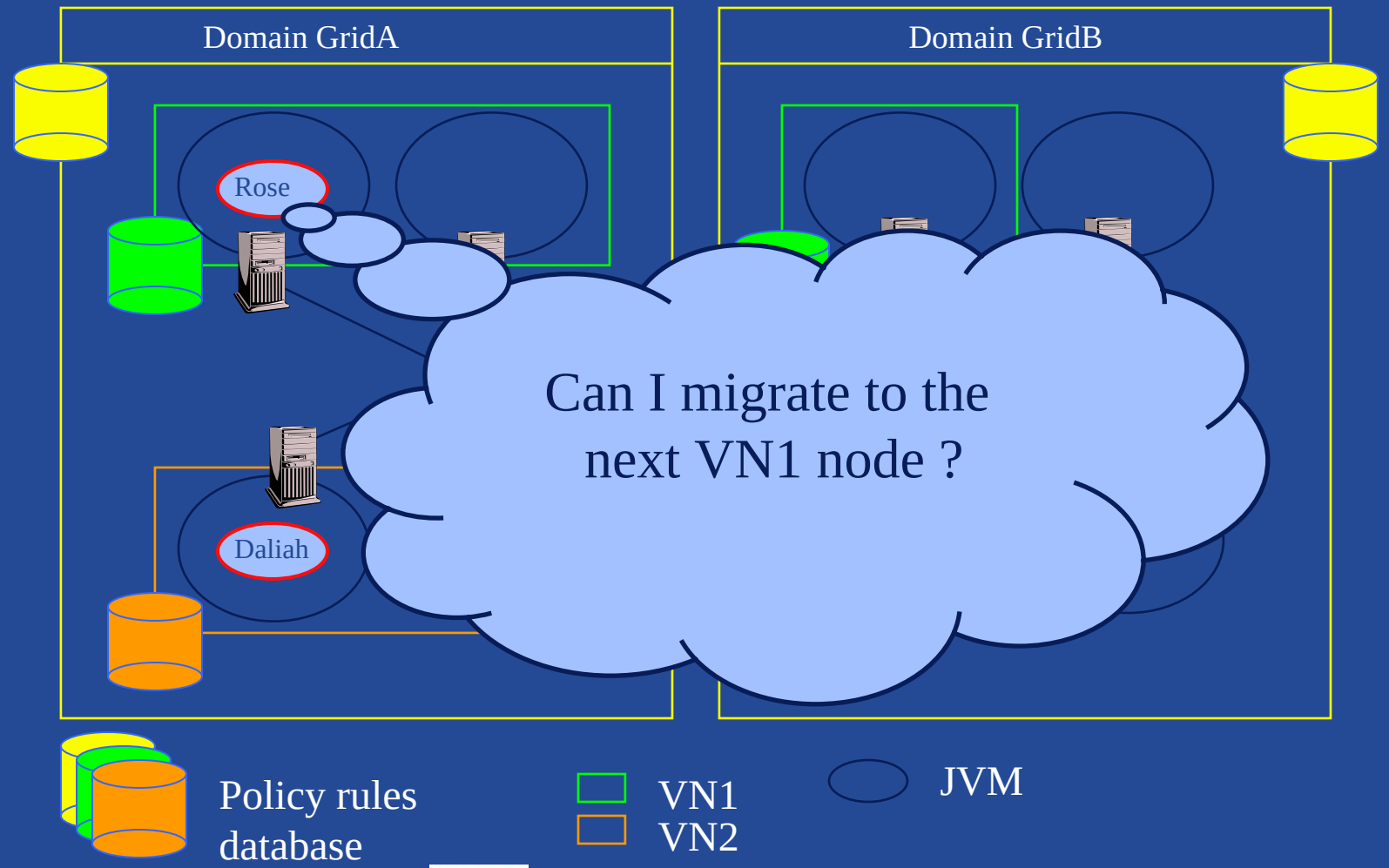

Domain GridA
Domain GridB
Policy rules database
VN1
VN2
JVM

Example


Domain GridA
Domain GridB
Rose
Daliah
Policy rules database
VN1
VN2
JVM

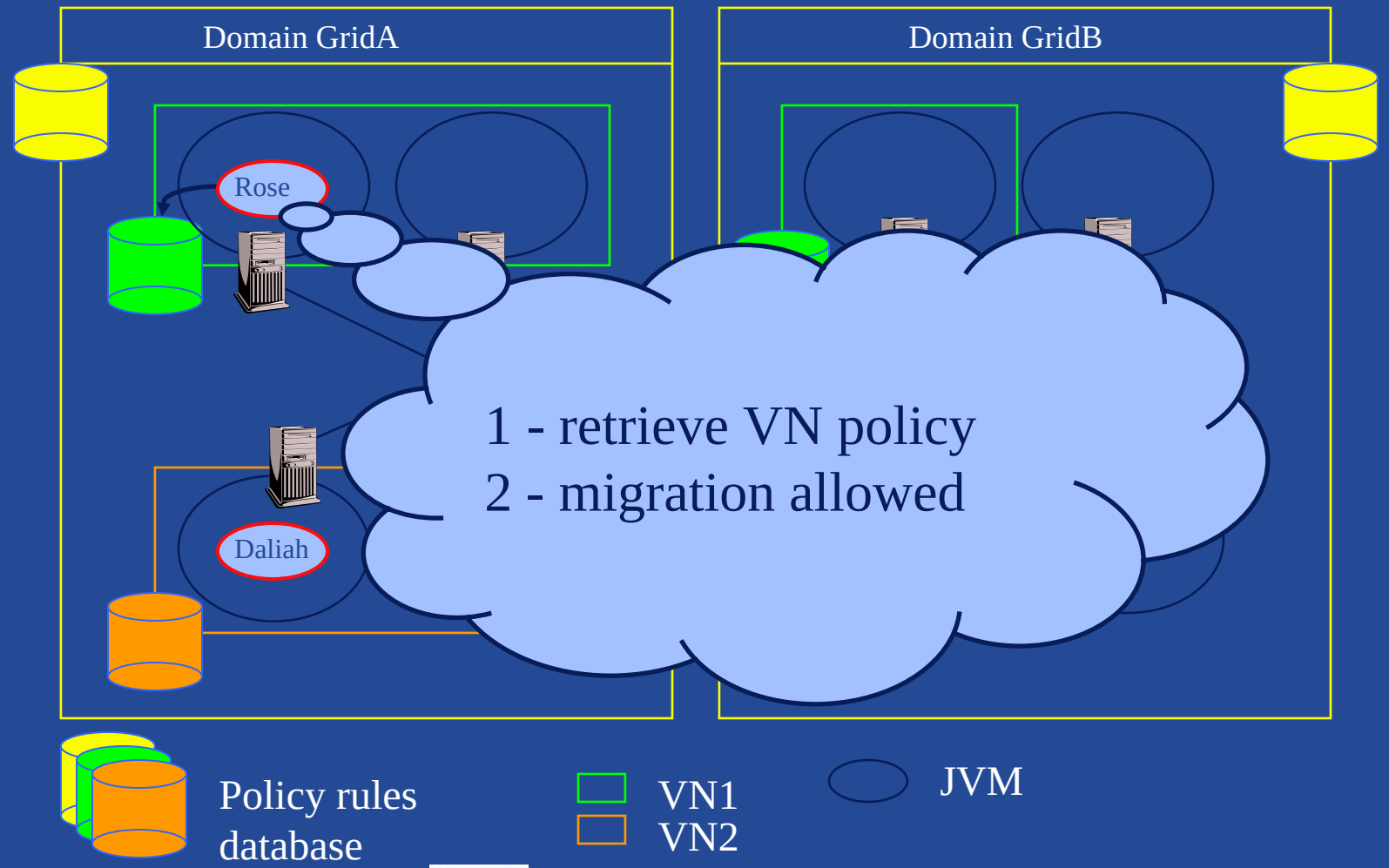
Example

Migration :
- same VN
- same domain



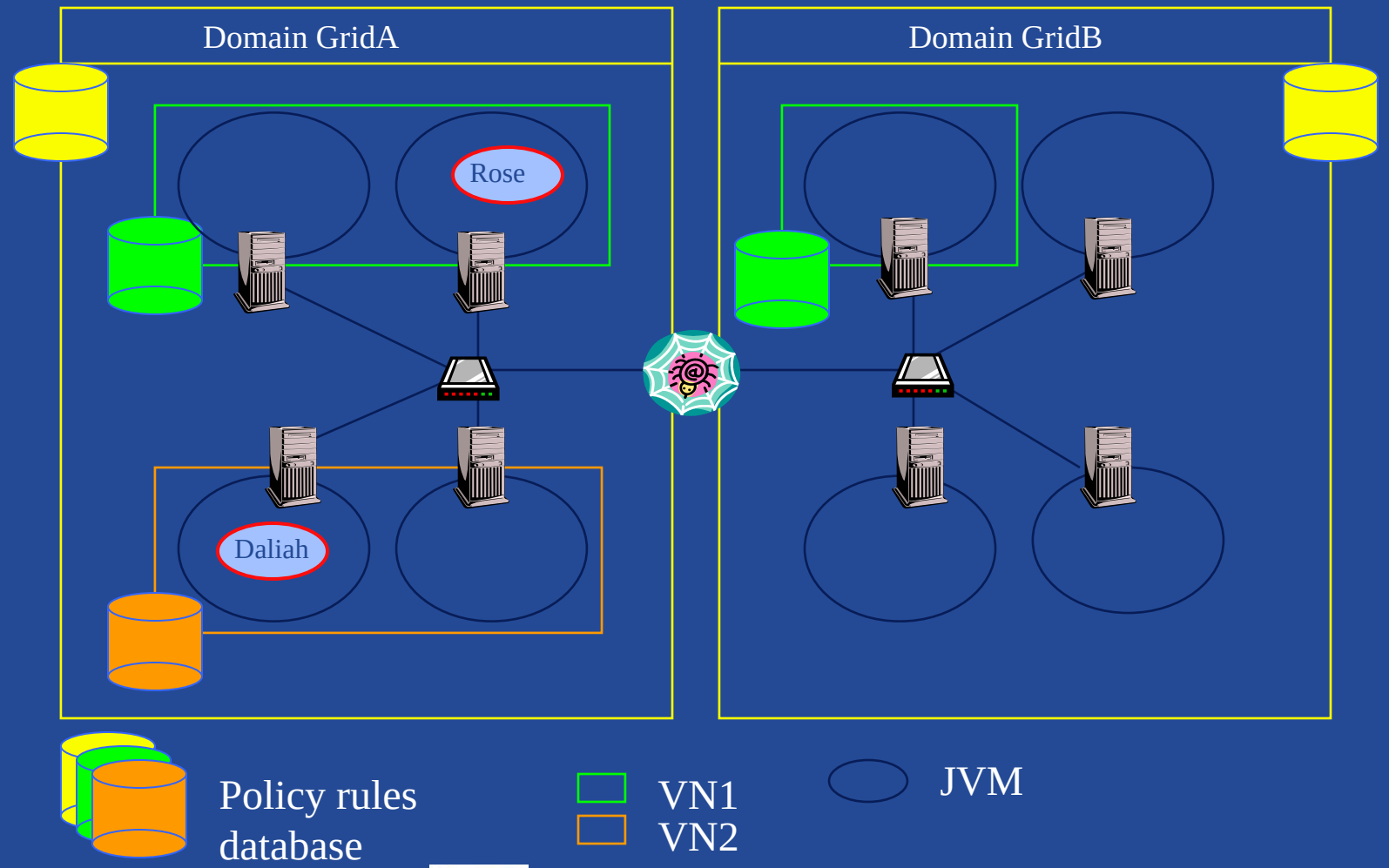
Example

Migration :
- same VN
- same domain



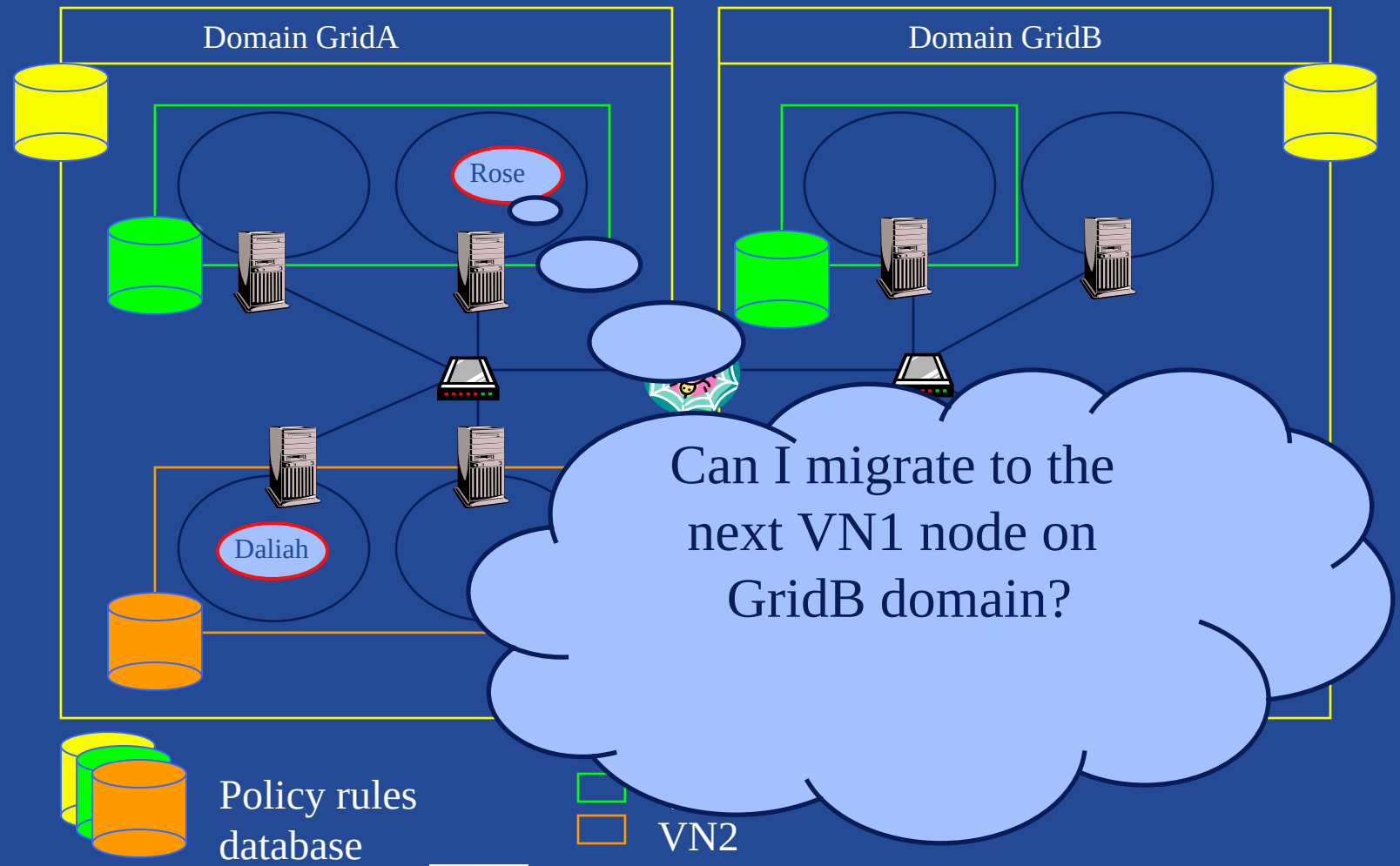
Example

Migration :
- same VN
- same domain



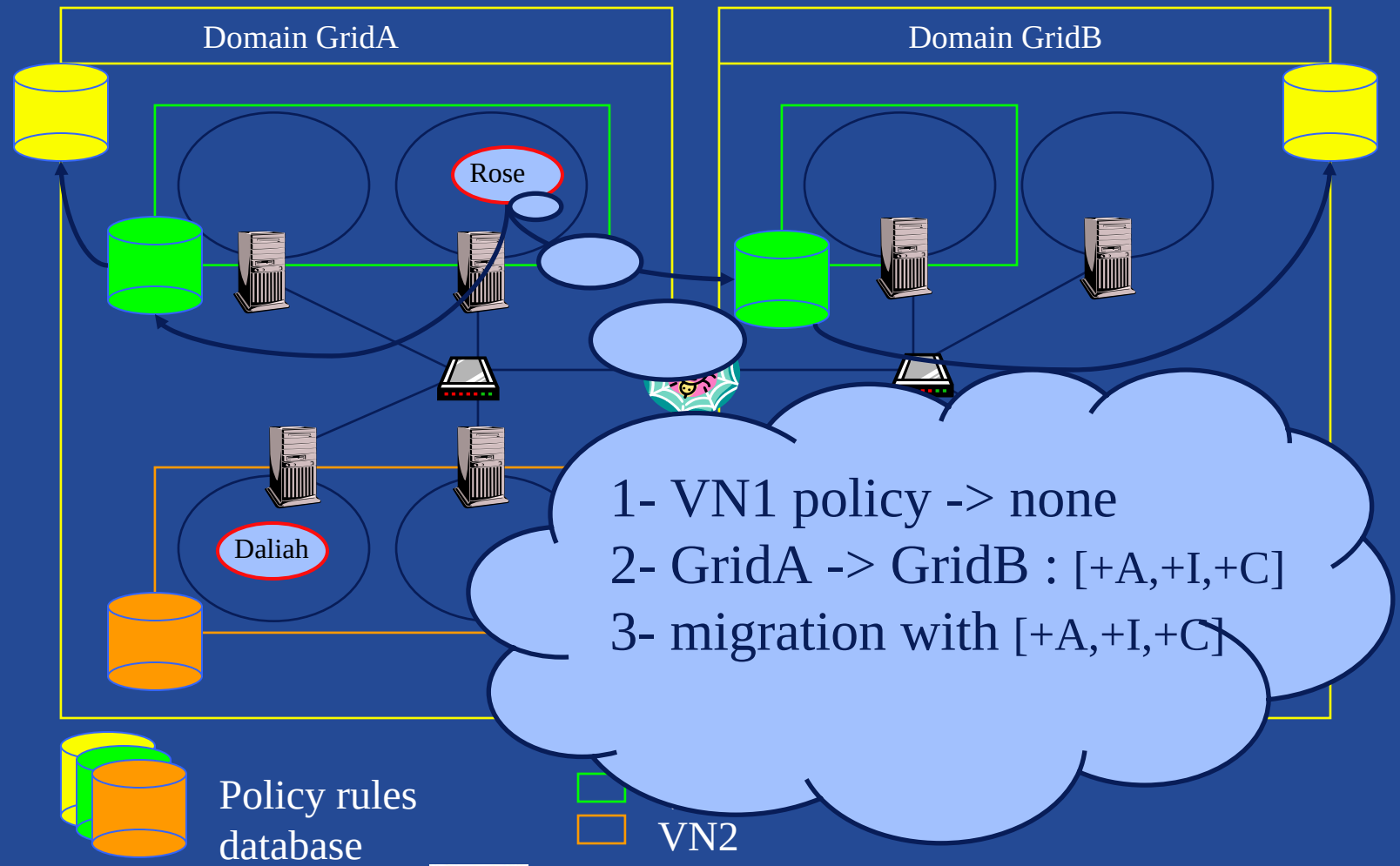
Example

Migration :
- same VN
- other domain



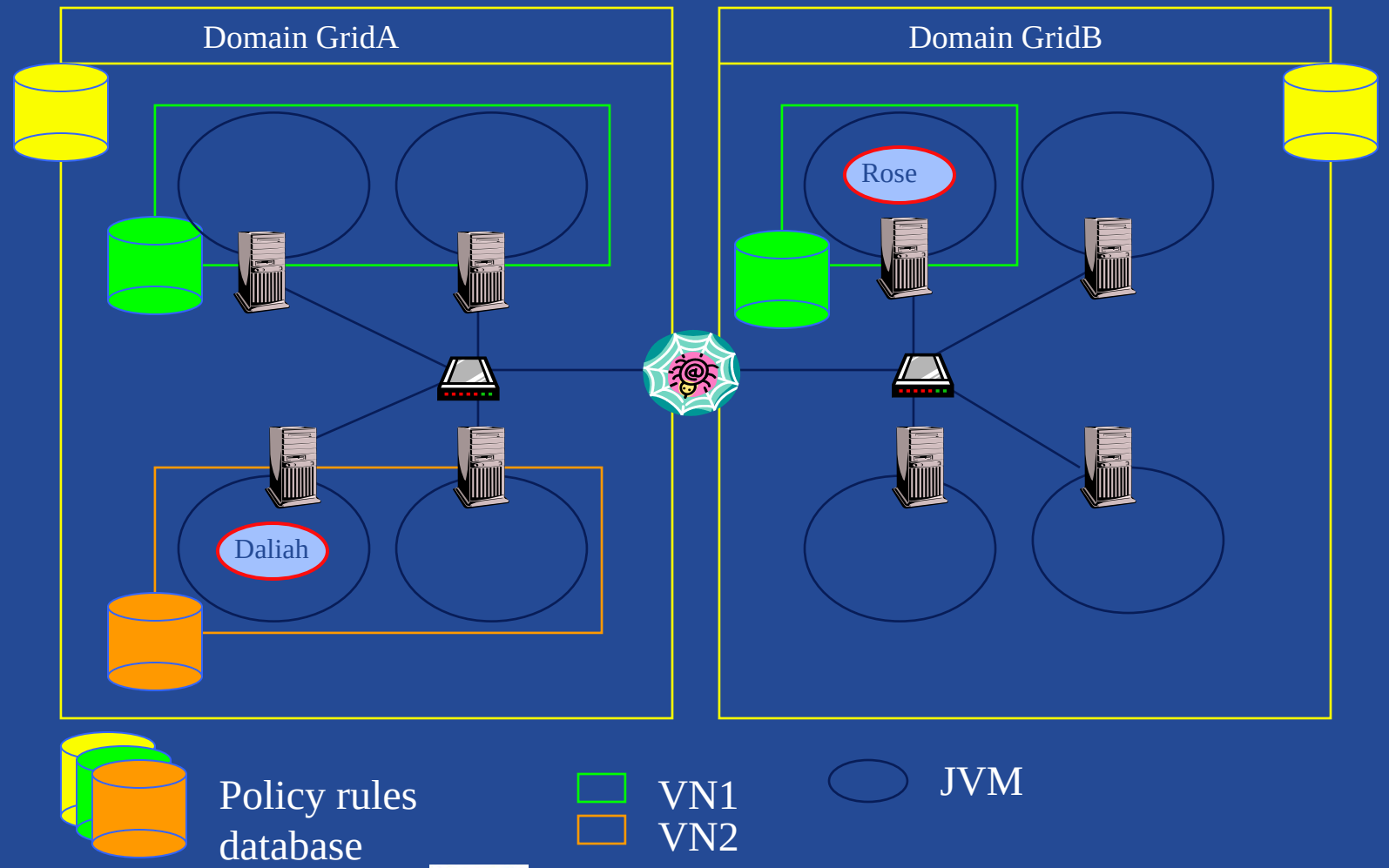
Example

Migration :
- same VN
- other domain

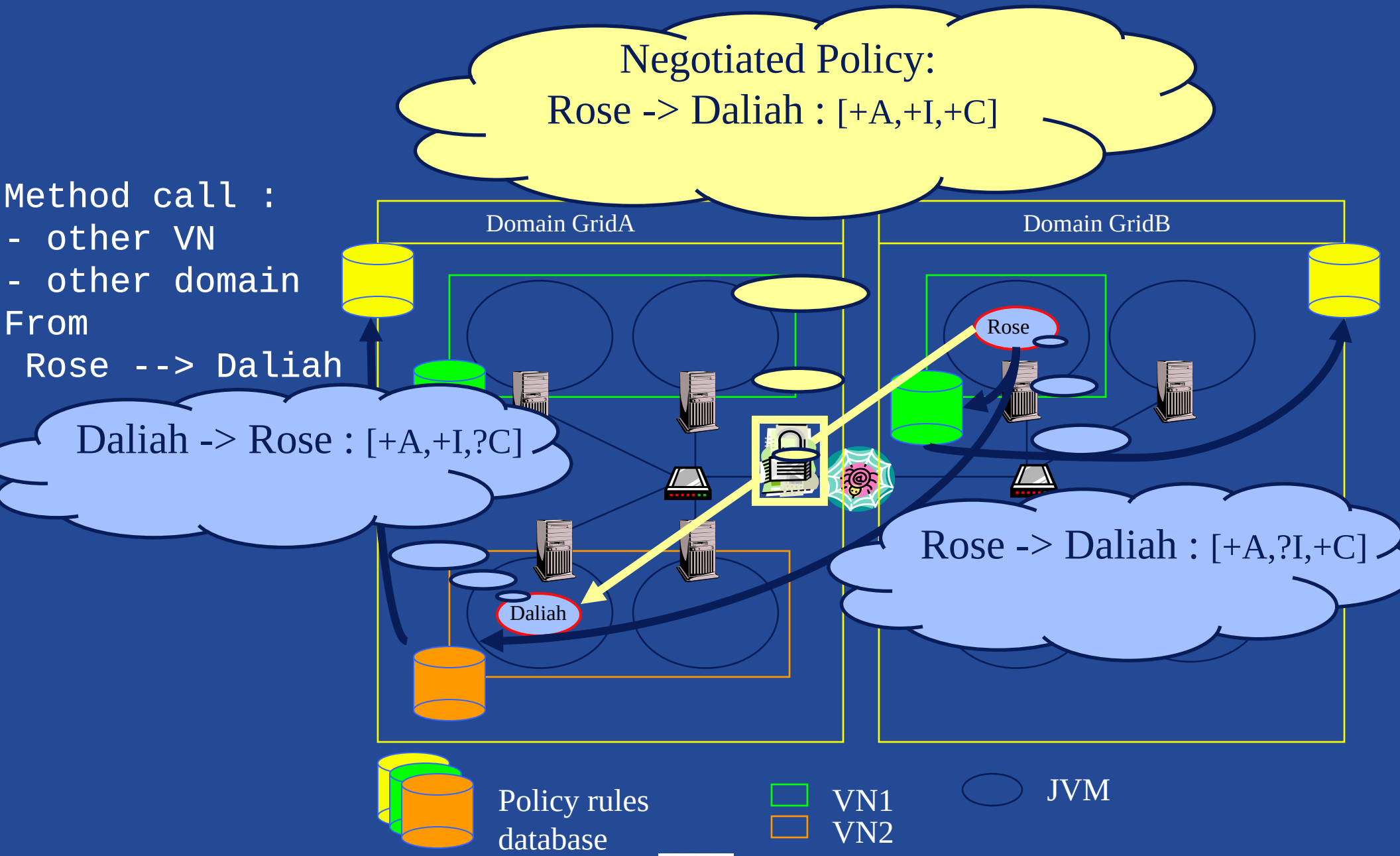


Example

Migration :
- same VN
- other domain



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



Adaptive Feature 4: Adaptive Security

Dynamic setting of the security attributes

Dynamic negotiation between different:

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

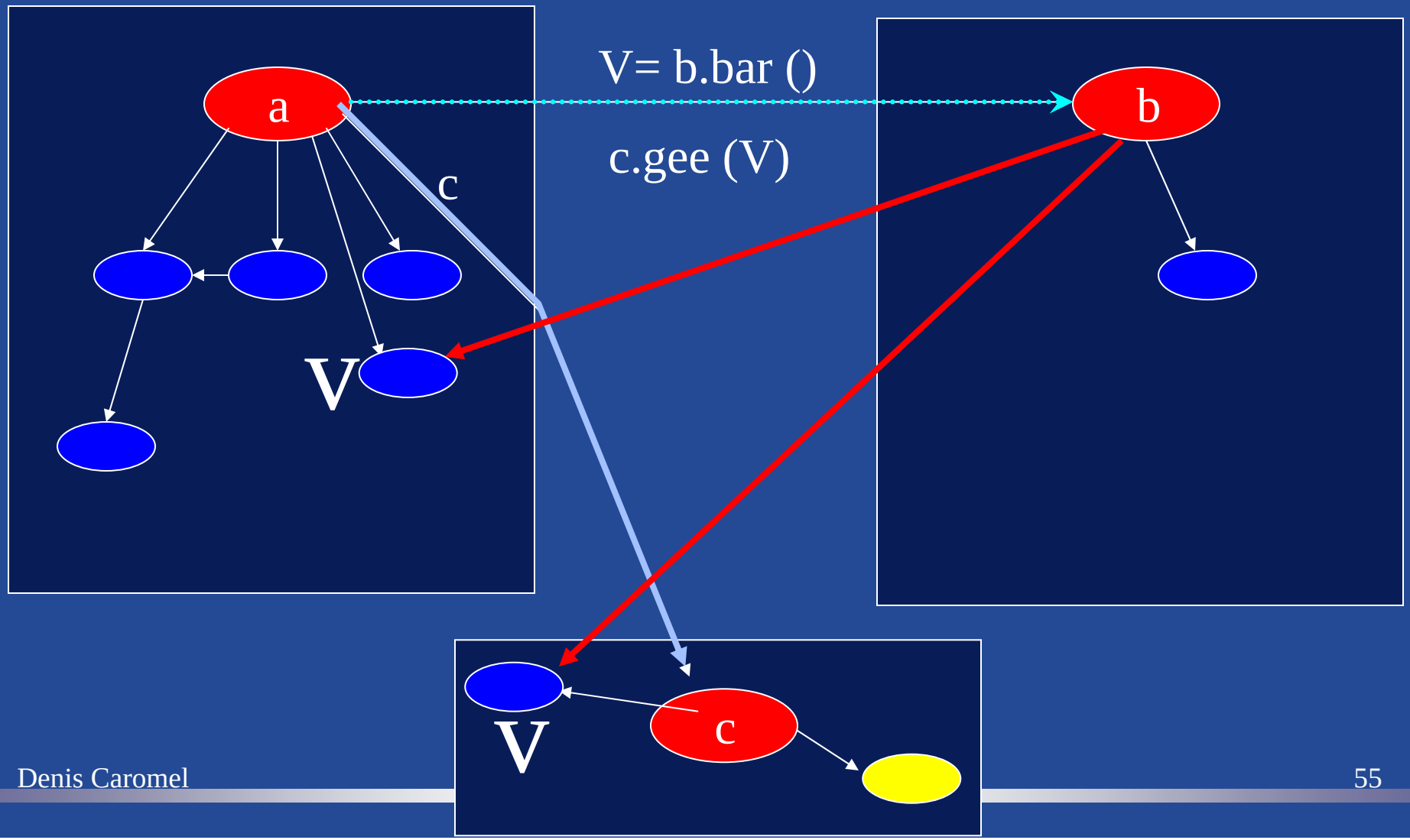
on JVMs on different Machines

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:

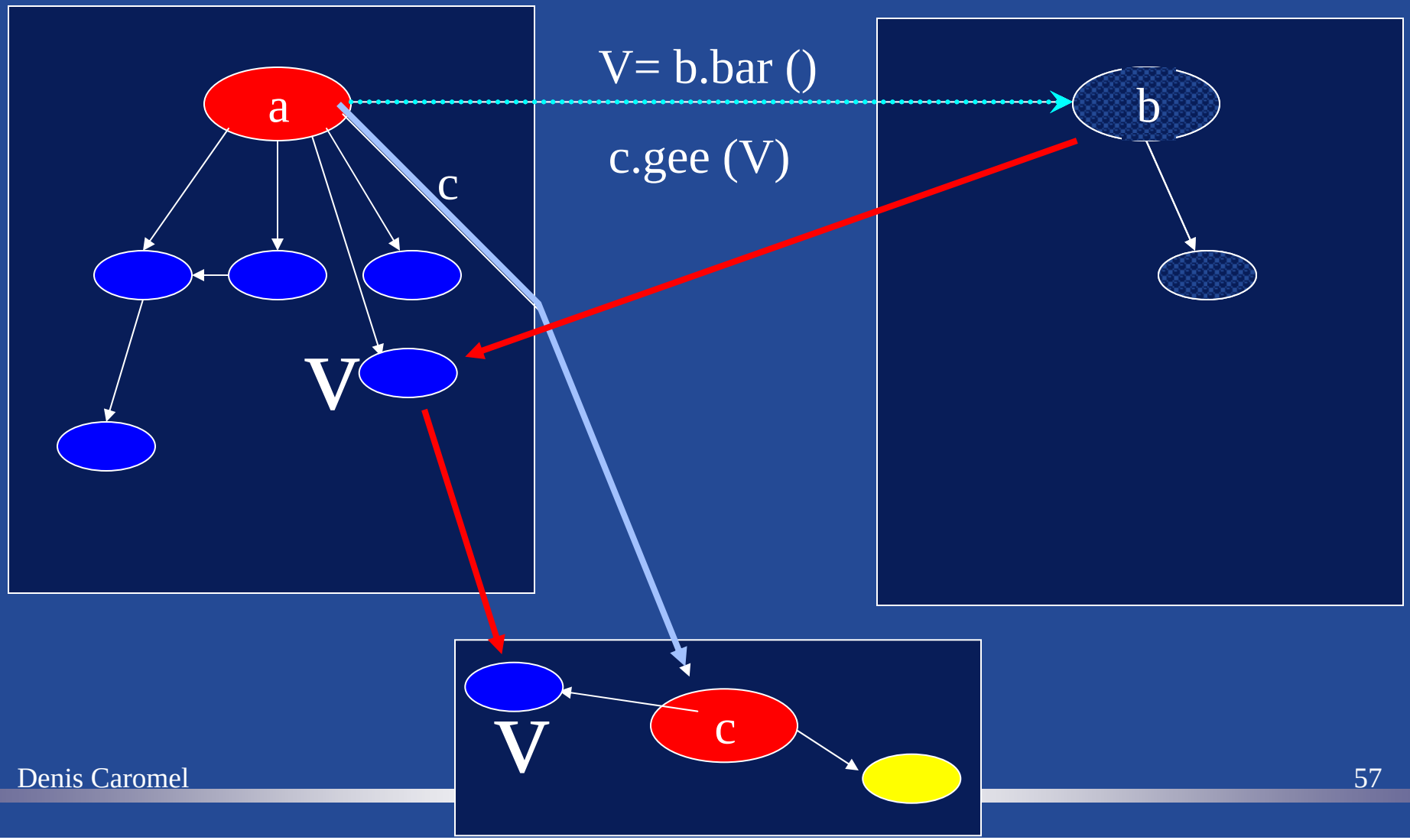
- Futures update any time between future computation and WbN

Lazy strategy:

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

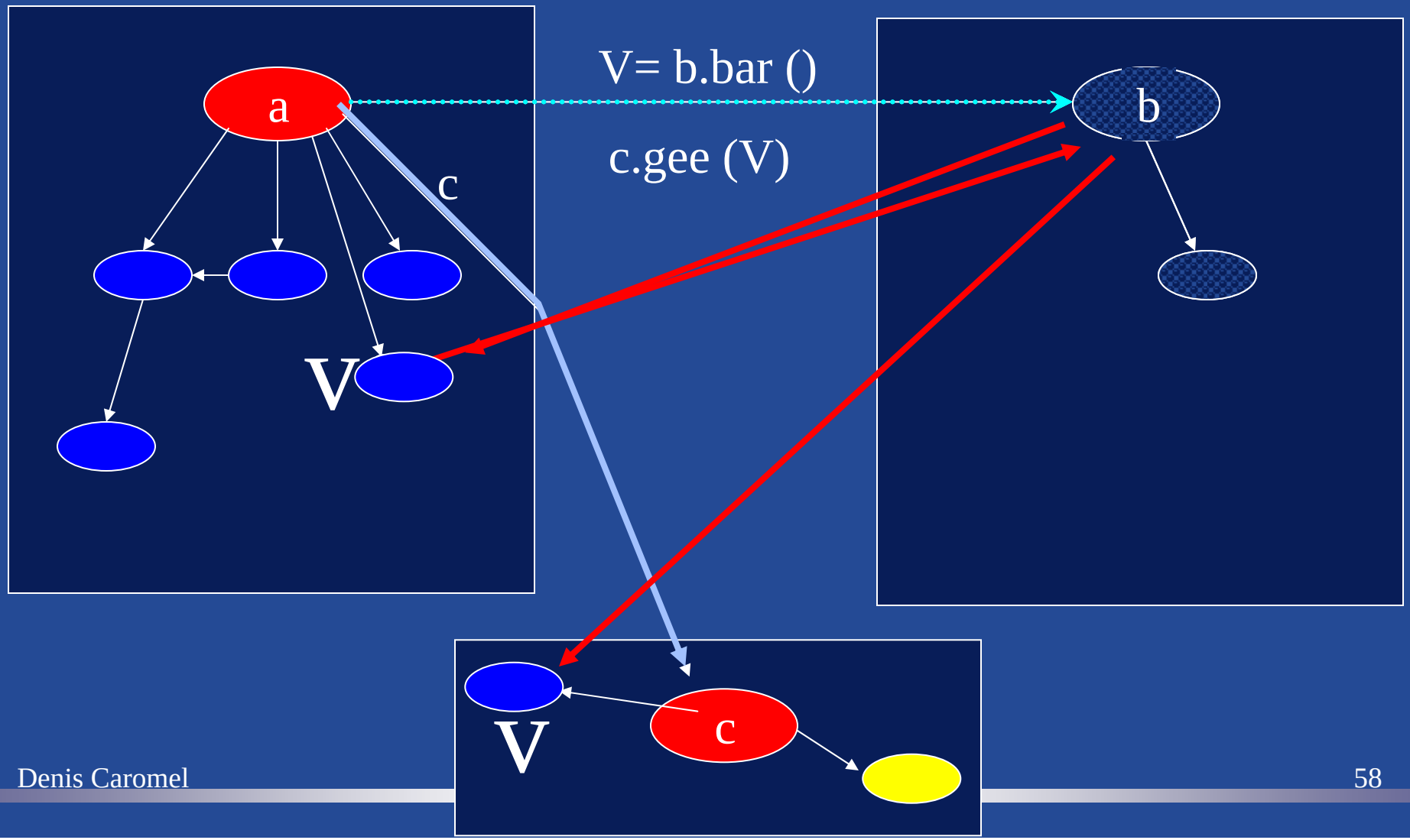
Wait-By-Necessity: Eager Forward Based

AO forwarding a future: will have to forward its value



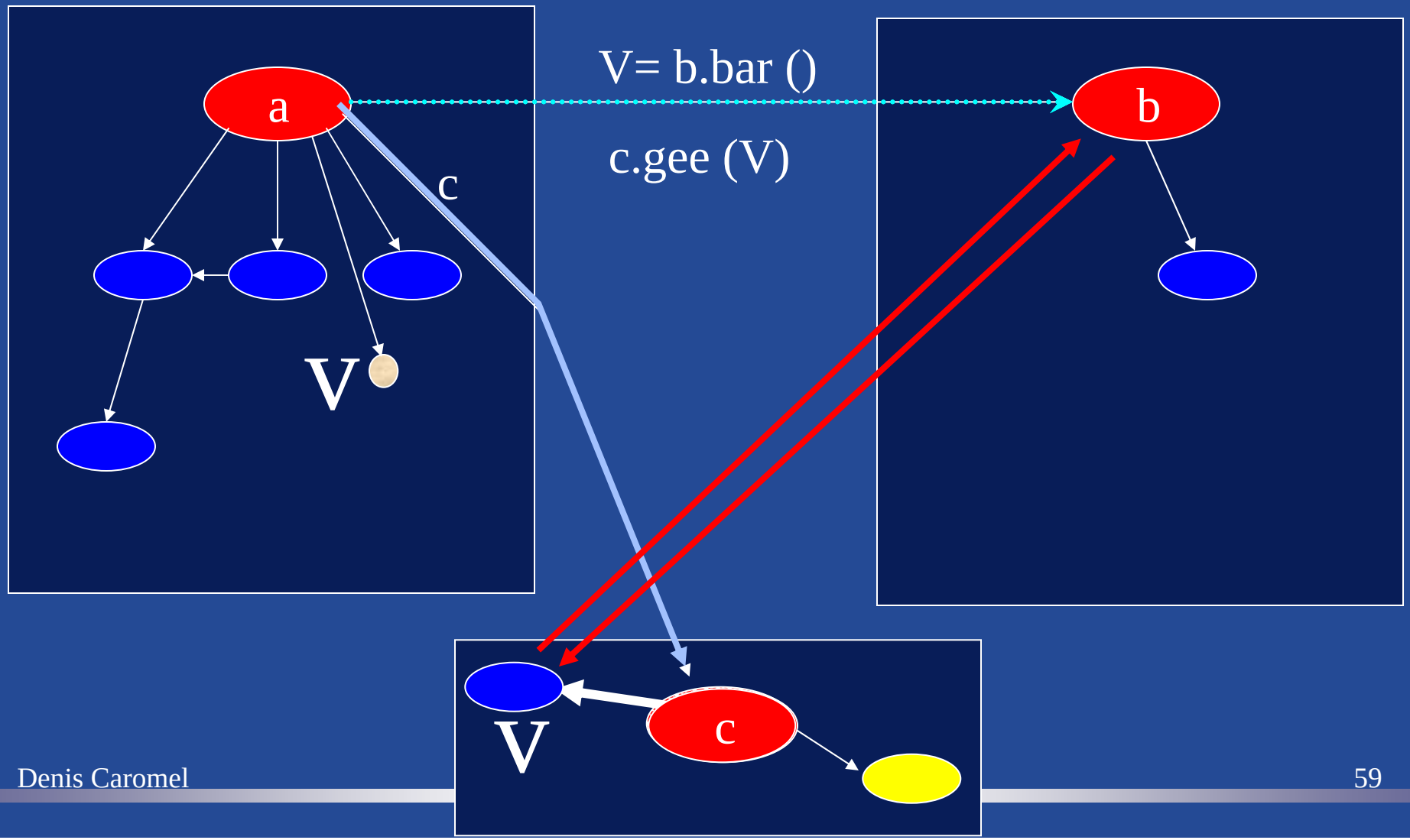
Wait-By-Necessity: Eager Message Based

AO forwarding a future: send a message



Wait-By-Necessity: Lazy Strategy

An Active Object requests a Future Value when needed



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 receivesS
- 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

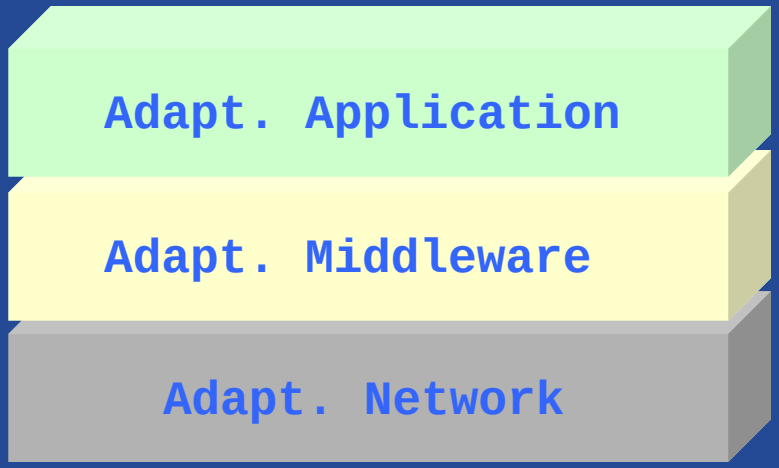
5 Adaptive/Parameterized features in ***ProActive***:

- 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 PassingS



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

Maladaptive Adaptive Grids !

TCP is an Adaptive Middleware

ProActive.ObjectWeb.org