GRID-RMI

High performance distributed objects for computing Grids

Projet logiciel PL-12

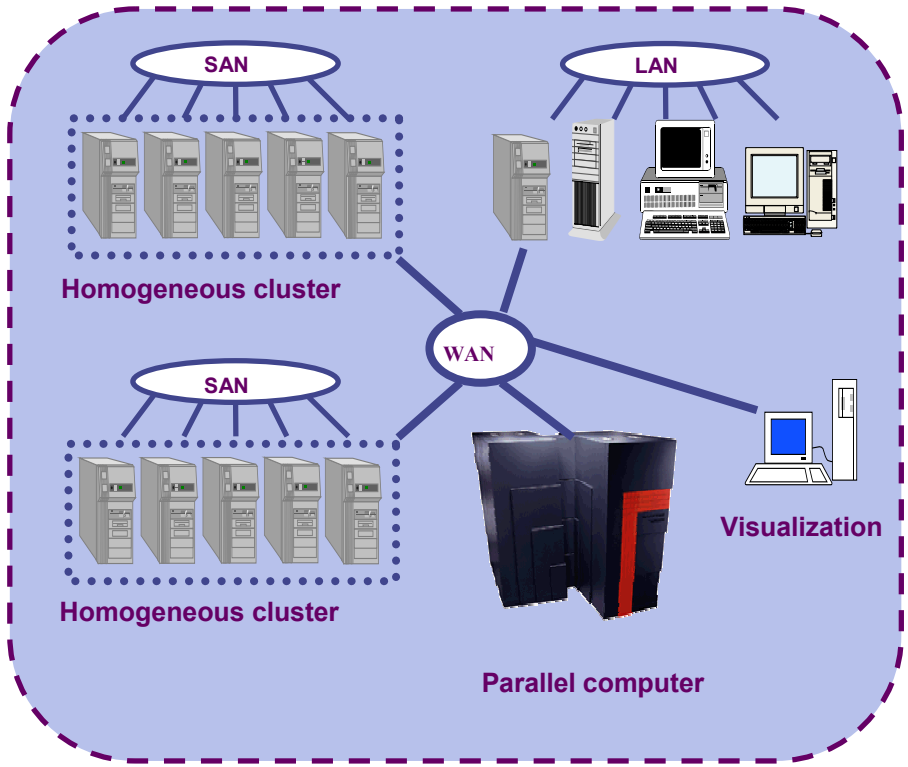
Action Concertée Incitative [ACI]
Globalisation des Ressources Informatiques et des Données [GRID]

Contents

- Context
- GRID RMI ACI
- Building blocs
  - Marcel/Madeleine
  - PadicoTM
- Programming model
  - Do!
  - ProActive
  - PaCO++
  - OpenCCM

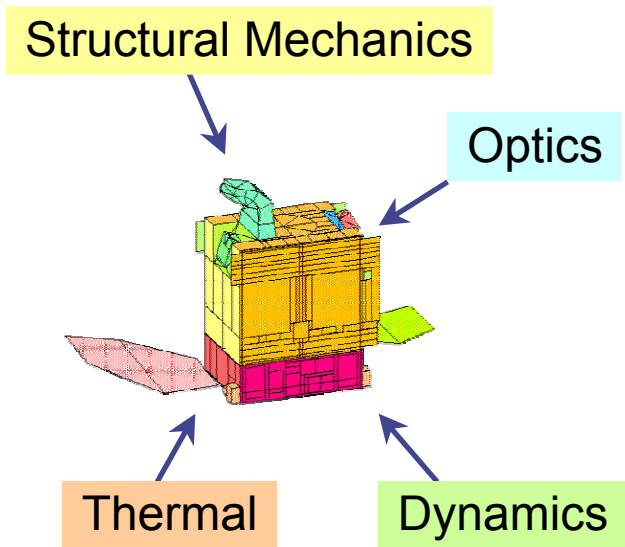
The Grid Computer

- Hierarchical networks
  - WAN : Internet, ATM, VTHD (2.5 Gbit/s), etc.
  - LAN : Ethernet
  - SAN : Myrinet, SCI
- Computing resources
  - Homogeneous clusters
  - Parallel computers
- Heterogeneity!



Grid Applications

- Parallel applications
  - Execution of *one* SPMD code
  - Grid seen as a virtual supercomputer
- Code coupling applications
  - Execution of several codes
    - Written with different languages
    - Using different communication paradigms (MPI, DSM, …)



Satellite design

Current views of the Grid

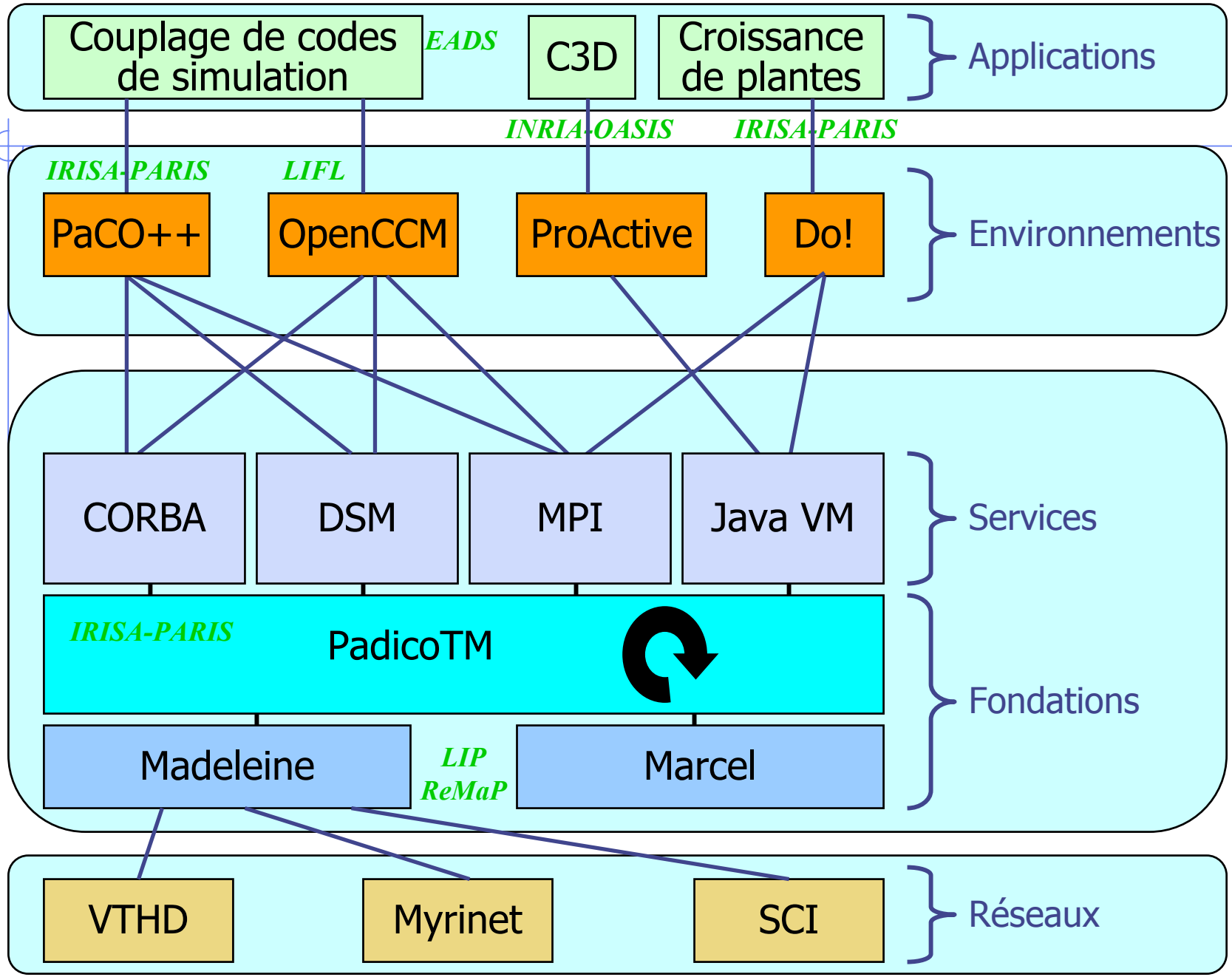
- Parallel programming view
  - Static network topology
  - Oriented towards “classical” parallel applications
  - Network heterogeneity difficult to handle
- Distributed programming view
  - Dynamic network topology
  - Oriented towards “client-server” applications
  - No support for high performance network interface

Some challenges to be solved

- Modern programming approach
  - RPC model is outdated
- Heterogeneity of networks, machines and languages
- High performance for cluster (SAN and zero copy)
- Asynchronous communications and parallelism
- Deployment and configuration at large scale
- Integration of system services
  - life cycle, activation, security, transactions, persistence, “real-time”, fault tolerance, load balancing, etc.

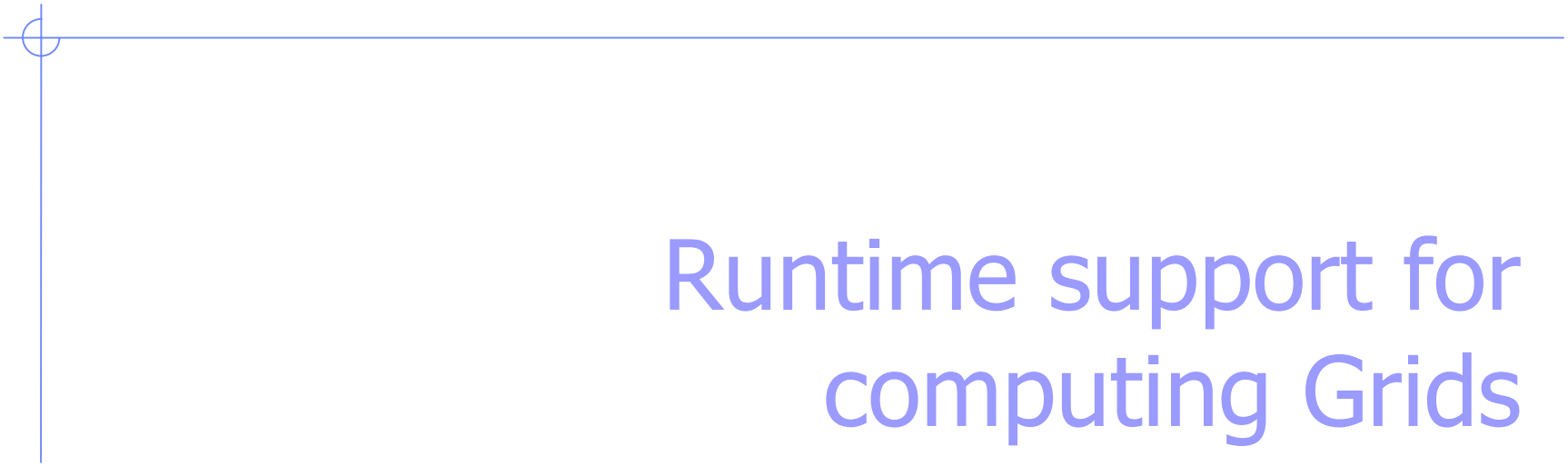
GRID-RMI goals

- Promote advanced programming models for the Grid
  - Integration of distributed and parallel models
- Modern programming approach
  - Distributed object: CORBA, JAVA
  - Software Component: CORBA Component Model
- Efficient network usage
  - All kinds of network: WAN, LAN et SAN
  - Transparently


Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

GRID RMI overview

- Software project
- Duration: 2 years (01/12/2001 -> 30/11/2003)
- Academic partners
  - LIP R. Namyst
  - LIFL J.-M. Geib, P. Merle
  - IRISA Y. Jégou, J.-L. Pazat, C. Pérez
  - INRIA-Sophia/I3S F. Baude, D. Caromel
- Industrial partner
  - EADS G. Alléon



Runtime support for computing Grids

Raymond Namyst

Projet ReMaP
LIP, ENS Lyon

µPM2

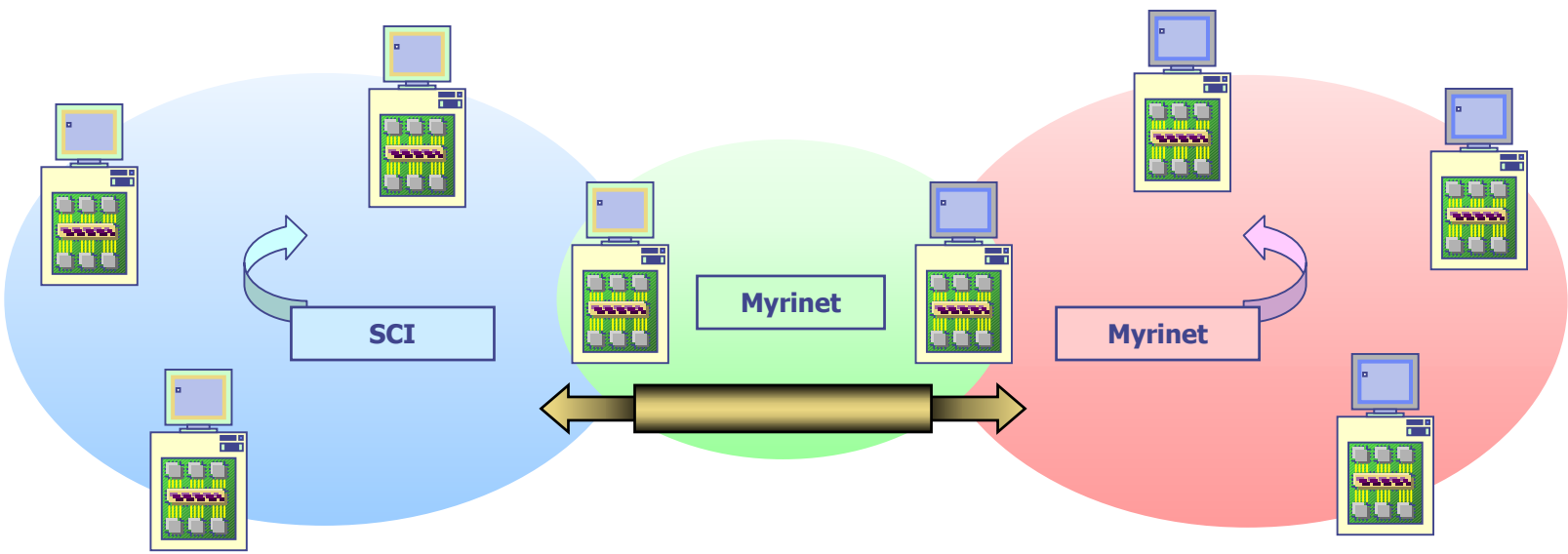
- µPM2 = Marcel + Madeleine
- Network library: Madeleine
  - Portability across several network kinds : Myrinet, SCI, VIA…
  - High performance: low latency, high bandwidth
- Multi-threading library: Marcel
  - Multi-threading in user space
  - Cooperates with Madeleine for good network reactivity

µPM2 and computing Grids

- Communications
  - Support for multi-cluster configurations
  - “low level” routing
  - Deployment
- Multithreading and reactivity
  - Support at the scheduler level
  - Aggregation of polling requests
  - Adaptation of polling frequencies
- Performance analysis
  - Understanding performances on SMP machines
  - Collecting accurate traces
  - Interfacing with existing tools

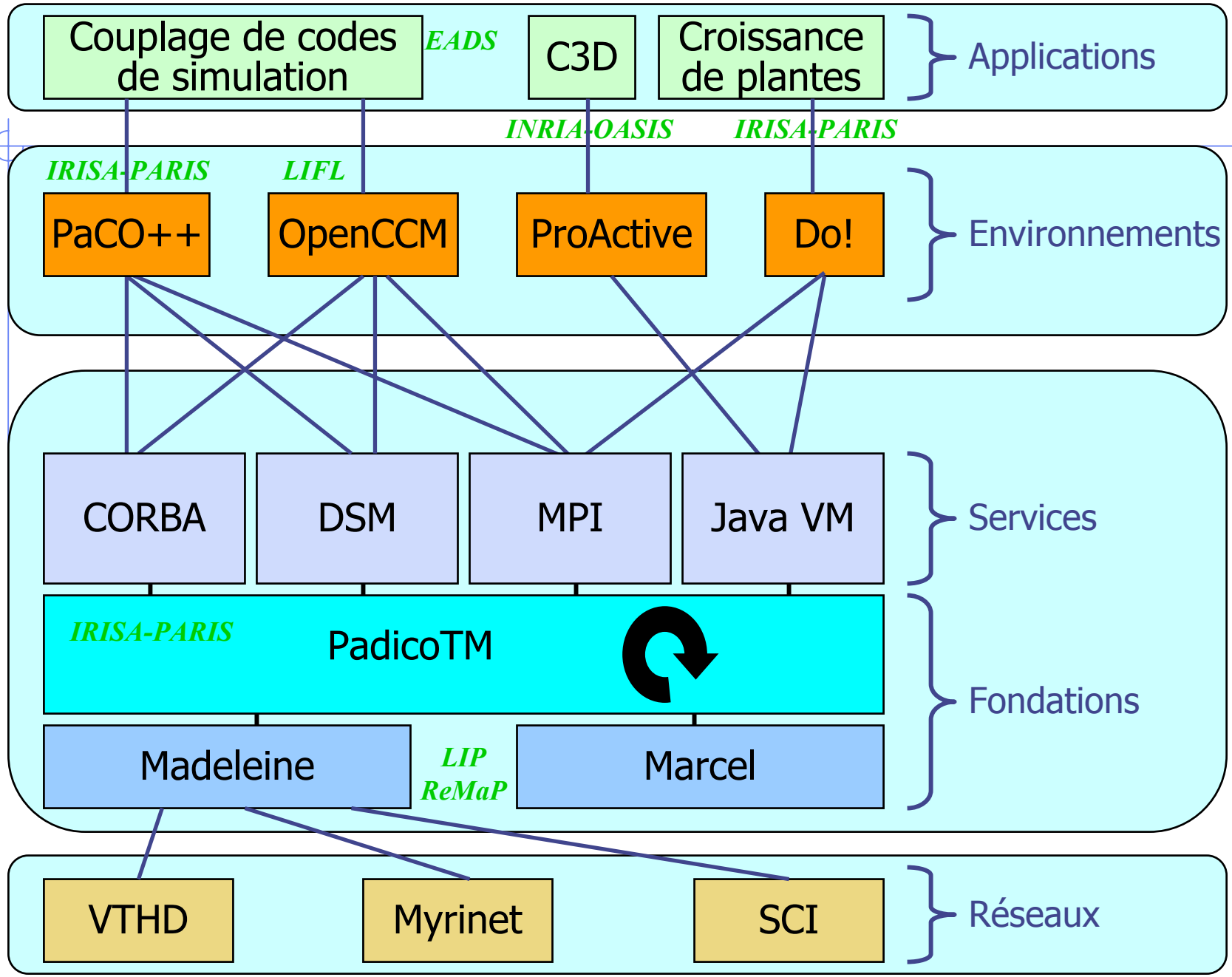
Communications for cluster of clusters

- Goals
  - Efficiency: High bandwidth between remote nodes (≠ PACX-MPI, MPICH-G2)
  - Transparent routing of messages
  - Deployment
- Bandwidth > 40 MB/s & 50 MB/s (max: 66 MB/s)



Multithreading and reactivity

- Active polling
  - Network API does not provide blocking call
  - Problem
    - Frequency difficult to tune
    - Expensive if multiple "*pollers*"
- Interruptions
  - Higher reactivity
  - LinuxActivations
    - Critical interruption path->user scheduler
- Proposition
  - Uniform interface to manage events
    - Scheduler support


Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

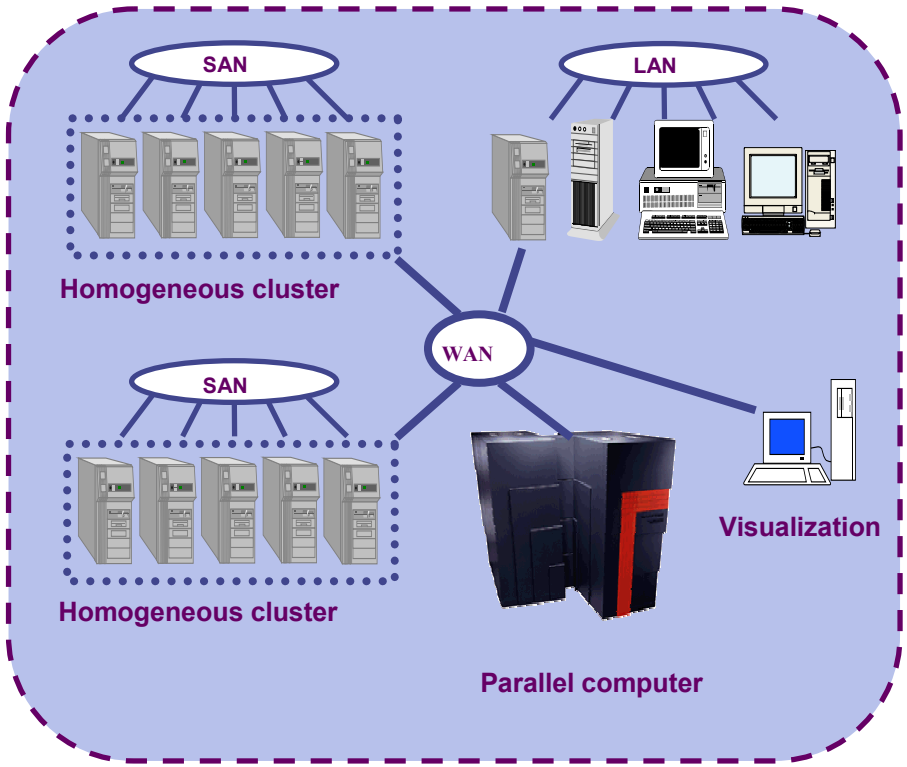
PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes

Alexandre Denis Christian Pérez
Thierry Priol André Ribes

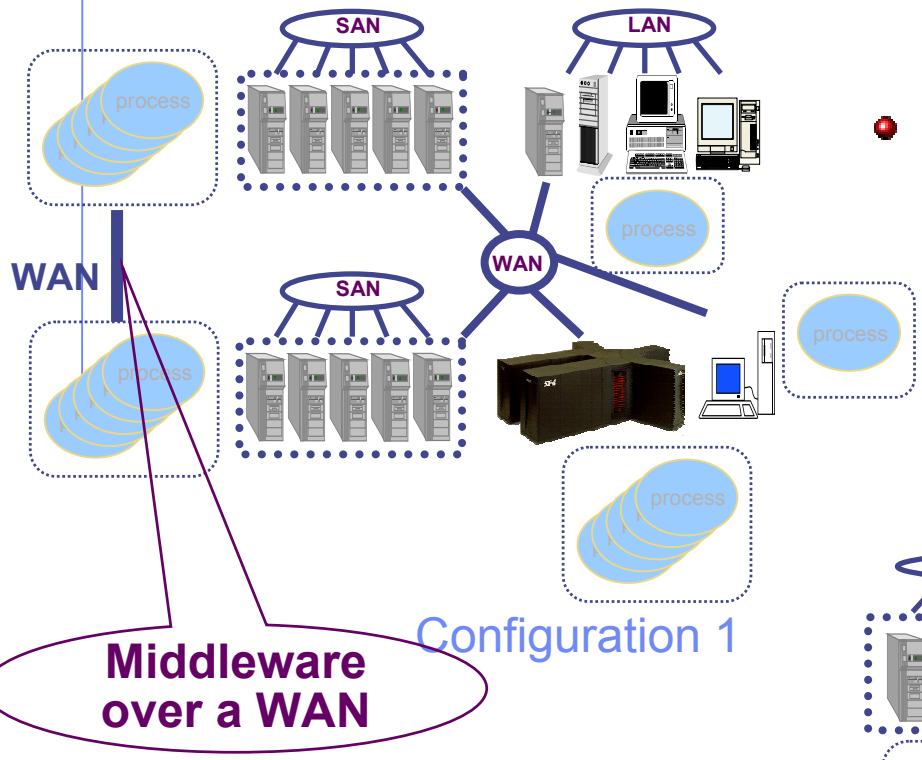
Projet PARIS
IRISA/INRIA, Rennes
http://www.irisa.fr/paris/Padico/

The Grid Computer

- Hierarchical networks
  - WAN : Internet, ATM, VTHD (2.5 Gbit/s), etc.
  - LAN : Ethernet
  - SAN : Myrinet, SCI
- Computing resources
  - Homogeneous clusters
  - Parallel computers
- Heterogeneity!

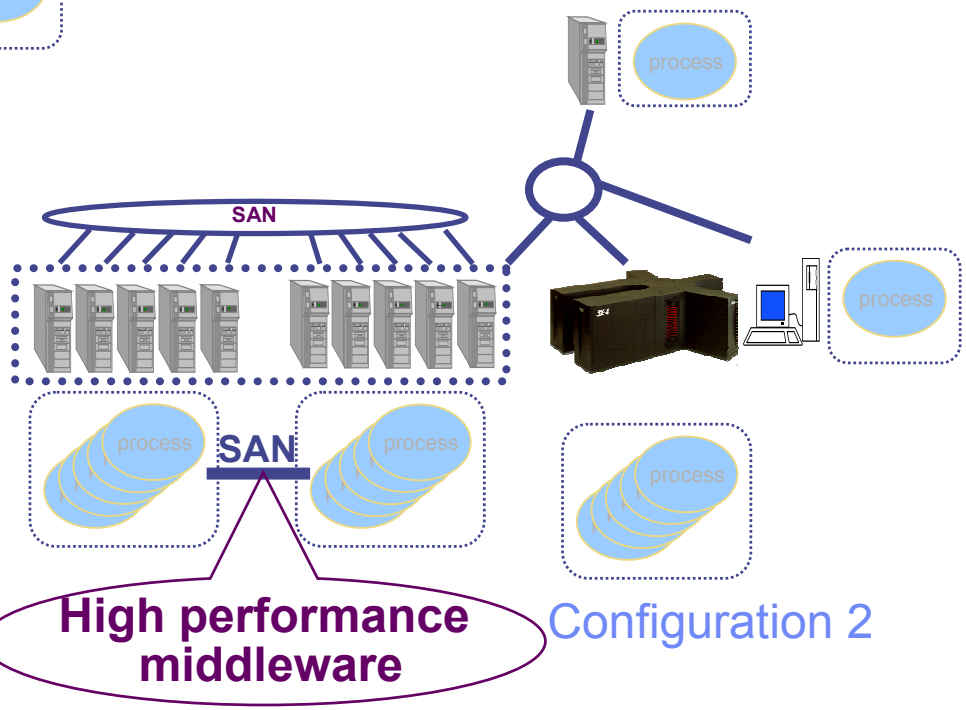


Mapping Objects Onto Resources



Configuration 1

- Deploy the ***same*** *code* using ***different*** *networking* resources

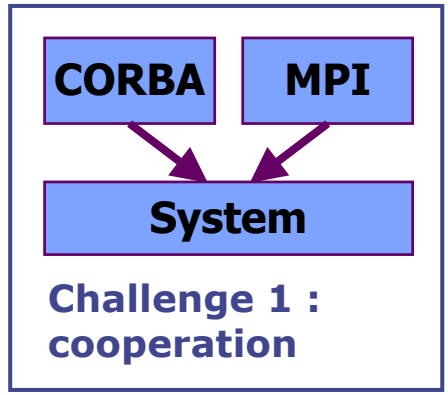


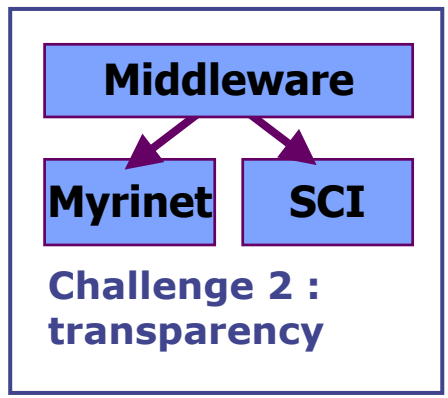
Configuration 2

- The middleware ***adapt itself*** to the available network

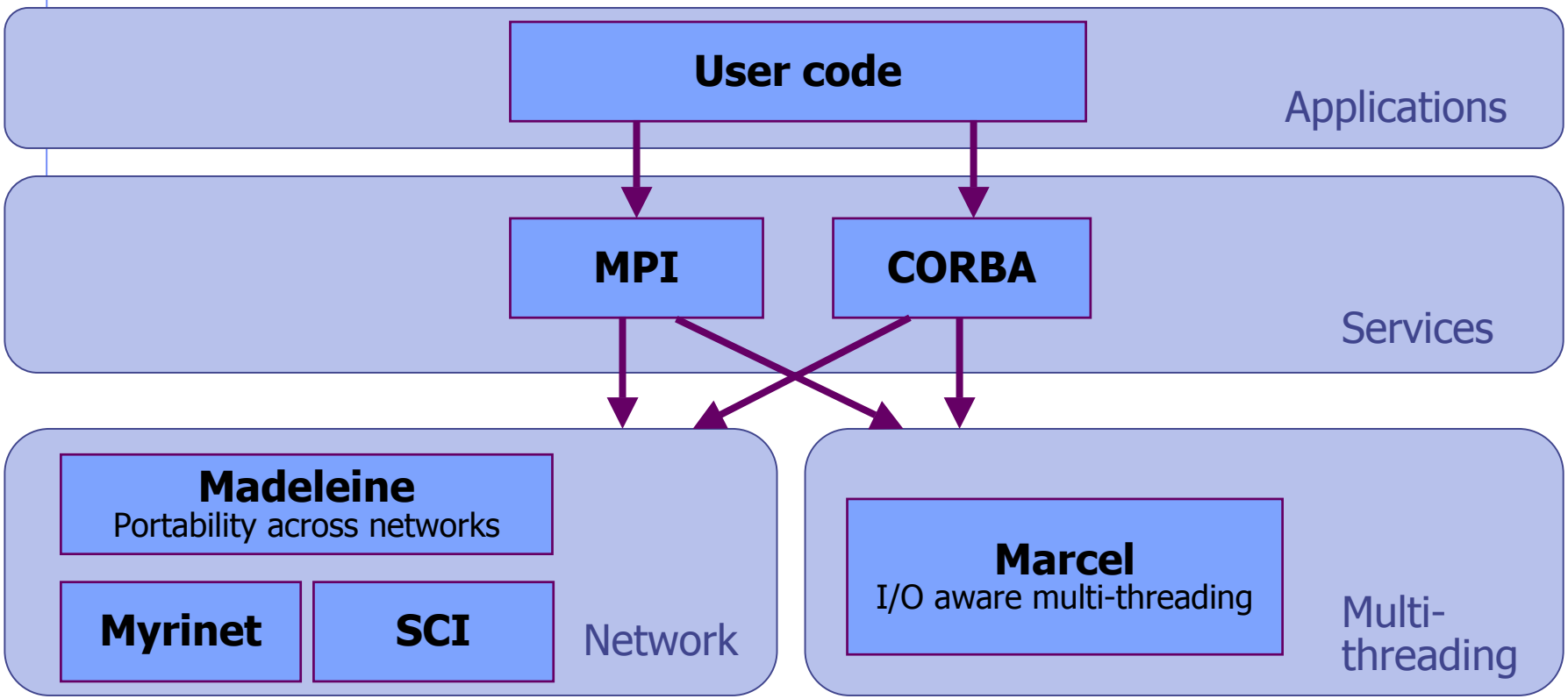
Challenges

- Several middleware systems ***at the same time***
  - Within the same process
  - Using the same network
- Use all resources ***transparently***
  - Each middleware system should be able to use *every network*
  - *Automatically* select the network
- ***High performance*** supplied to every middleware system

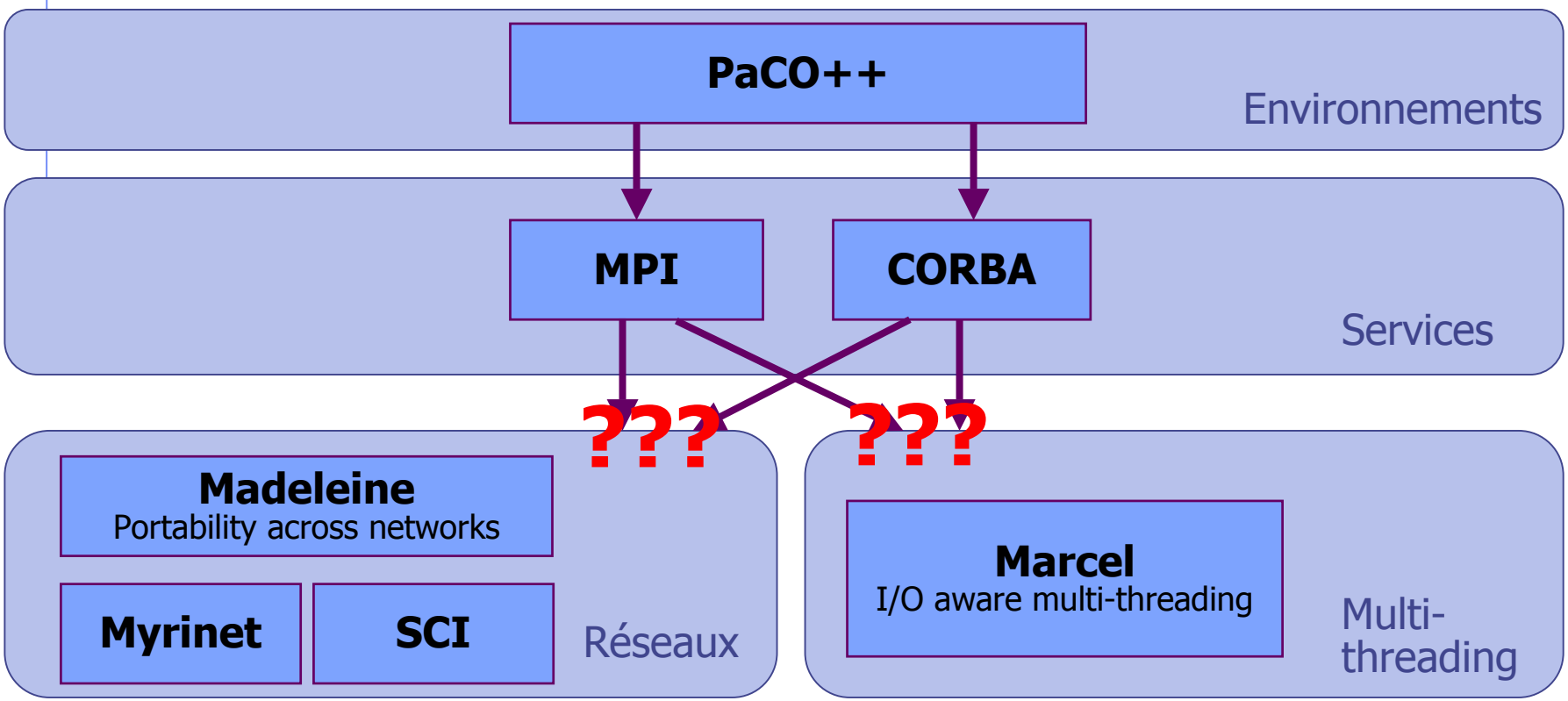




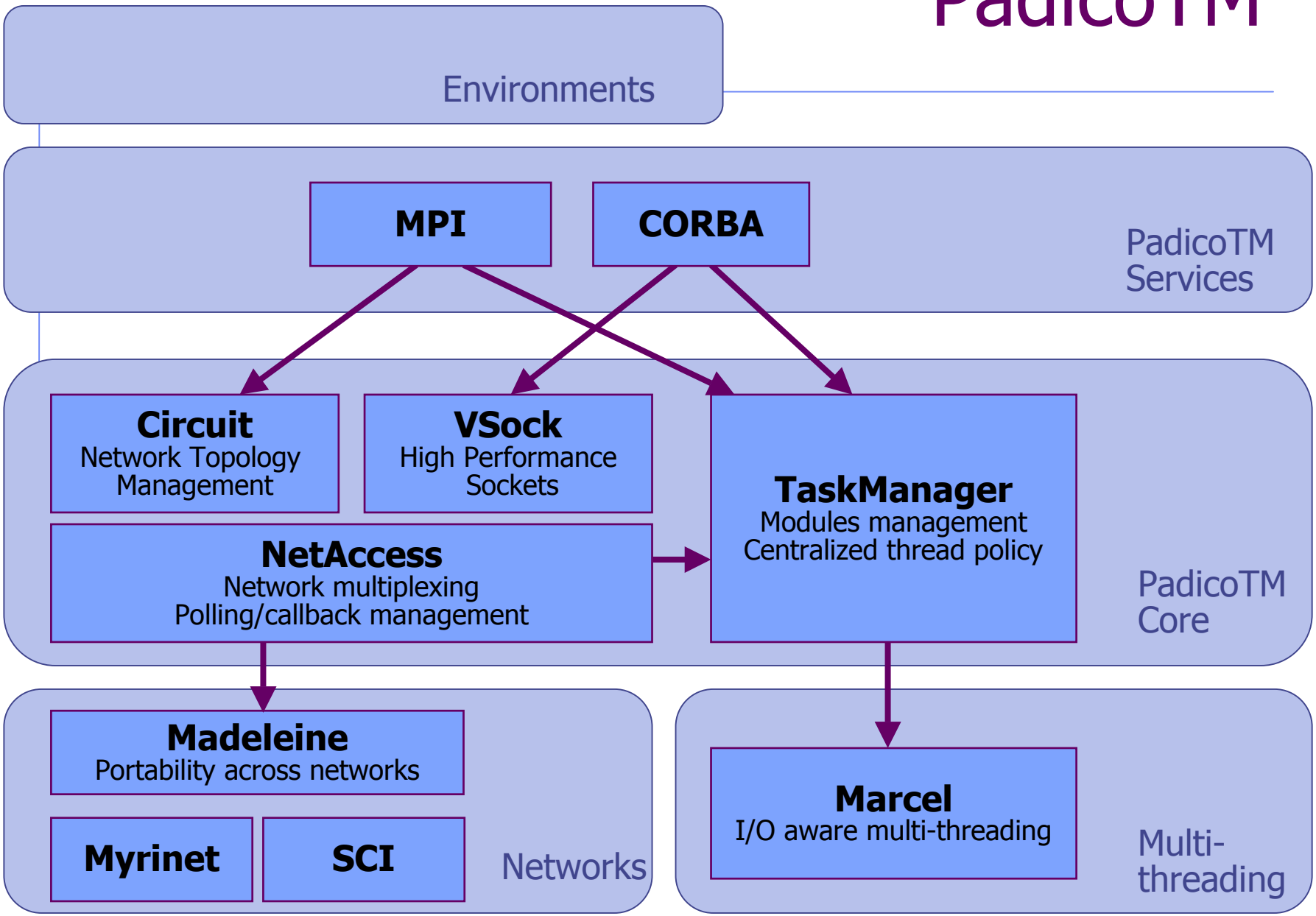
System Layer: Marcel and Madeleine



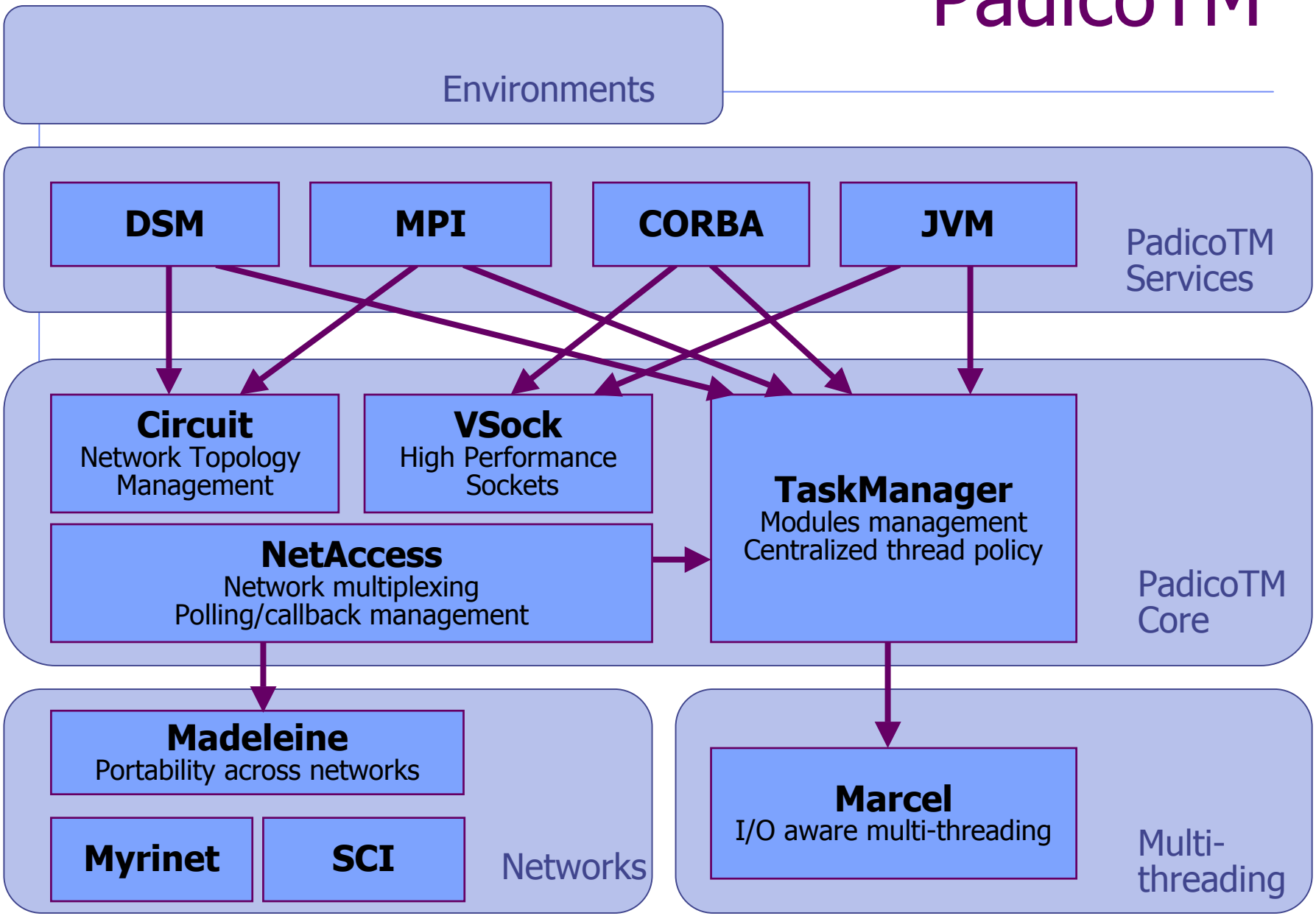
Access conflicts



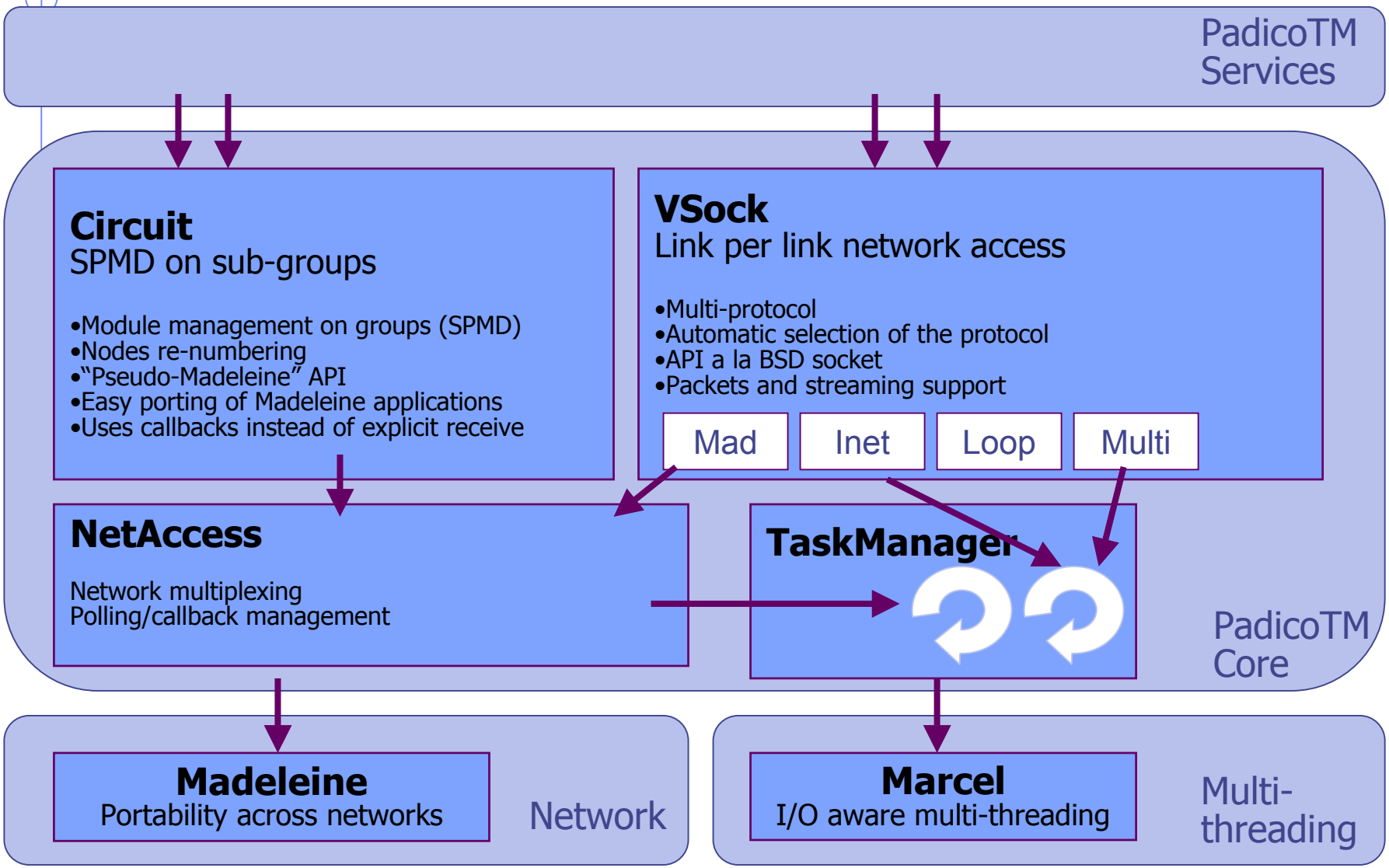
PadicoTM


Environments
MPI
CORBA
PadicoTM Services
Circuit
Network Topology Management
VSock
High Performance Sockets
NetAccess
Network multiplexing
Polling/callback management
TaskManager
Modules management
Centralized thread policy
PadicoTM Core
Madeleine
Portability across networks
Myrinet
SCI
Networks
Marcel
I/O aware multi-threading
Multi-threading

PadicoTM

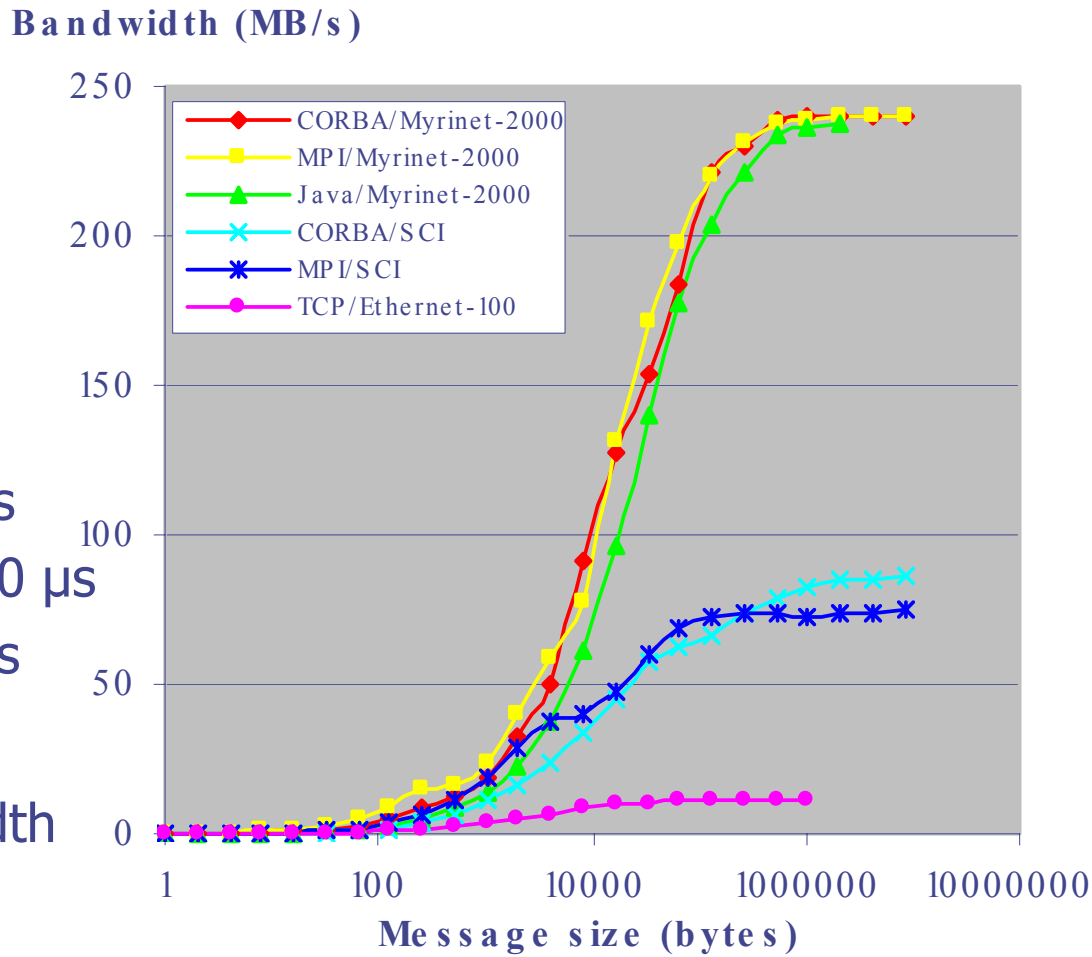

Environments
DSM
MPI
CORBA
JVM
PadicoTM Services
Circuit
Network Topology Management
VSock
High Performance Sockets
TaskManager
Modules management
Centralized thread policy
NetAccess
Network multiplexing
Polling/callback management
PadicoTM Core
Madeleine
Portability across networks
Myrinet
SCI
Networks
Marcel
I/O aware multi-threading
Multi-threading

Network Access details



PadicoTM Services Performance

- Bandwidth
  - Myrinet-2000 : 240MB/s
  - SCI : 86MB/s
  - 96% of the hardware bandwidth!
- Latency
  - CORBA/Ethernet : 150 µs
  - CORBA/Myrinet-2000 : 20 µs
  - MPI/Myrinet-2000 : 11 µs
- Improvement
  - For latency ***and*** bandwidth



PadicoTM services over Myrinet-2000, SCI and TCP/Ethernet-100

Current status

- Enhance deployment and resource management
- PadicoTM comprises:
  - PadicoTM core ✓
  - MPI ✓
  - CORBA
    - omniORB ✓
    - Mico/MT ✓
    - MicoCCM *being developed*
  - JVM *in tests*
  - DSM *to be done*
- Current release: 0.1.2 (July 2002)

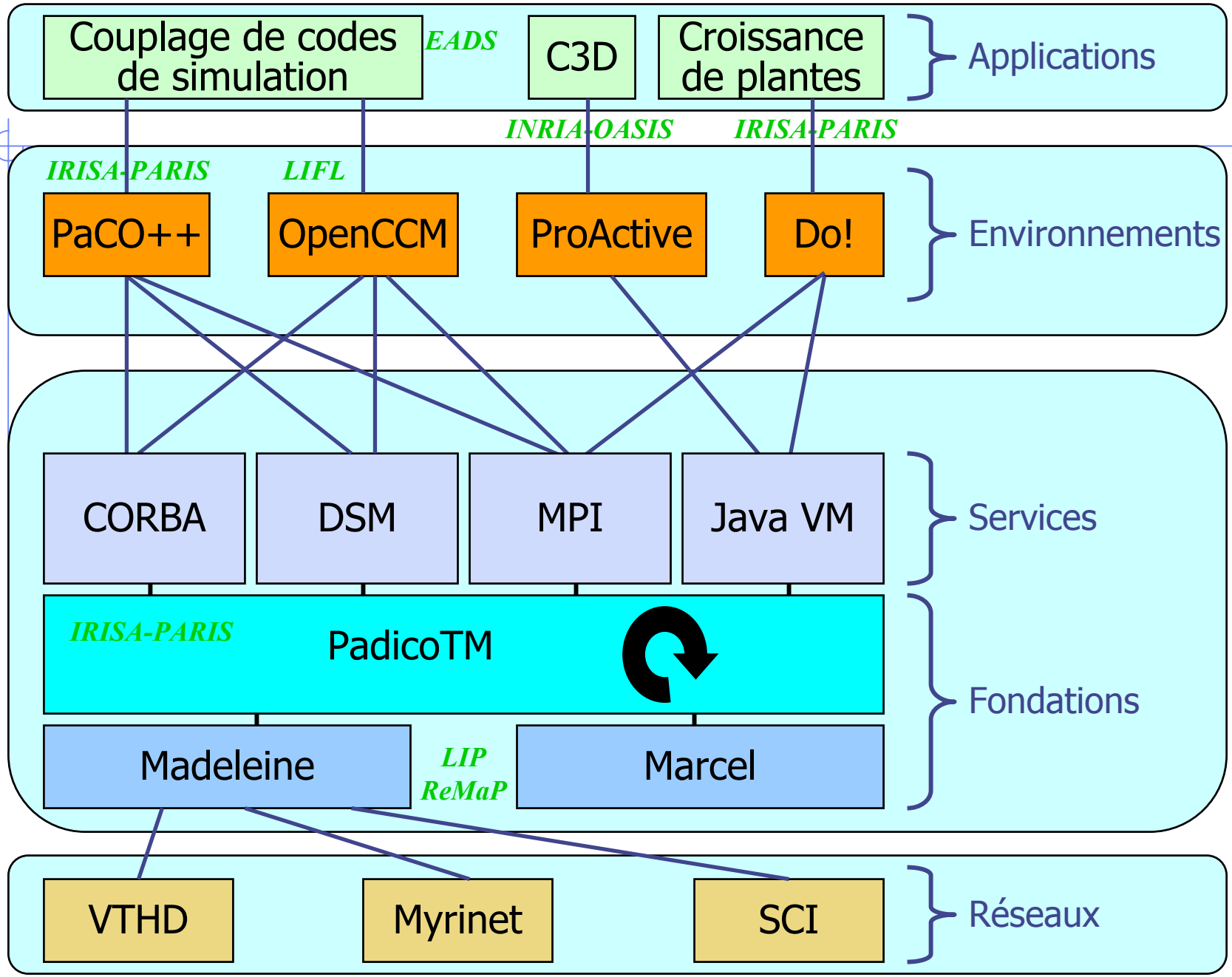
Conclusion

- PadicoTM: an open integration framework for communication middleware and runtimes
  - High performance supplied to every middleware systems
    - ➔High performance CORBA
    - ➔High performance Java sockets
    - ➔High performance MPI
  - Supports several middleware at the same time

➔Towards new programming models

- ➔Parallel CORBA Object: PaCO
- ➔Parallel CORBA Component Model: GridCCM

  - Transparency and interoperability


Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

Do !
Generation of distributed parallel programs within a object framework

Jean-Louis Pazat

PARIS

IRSA

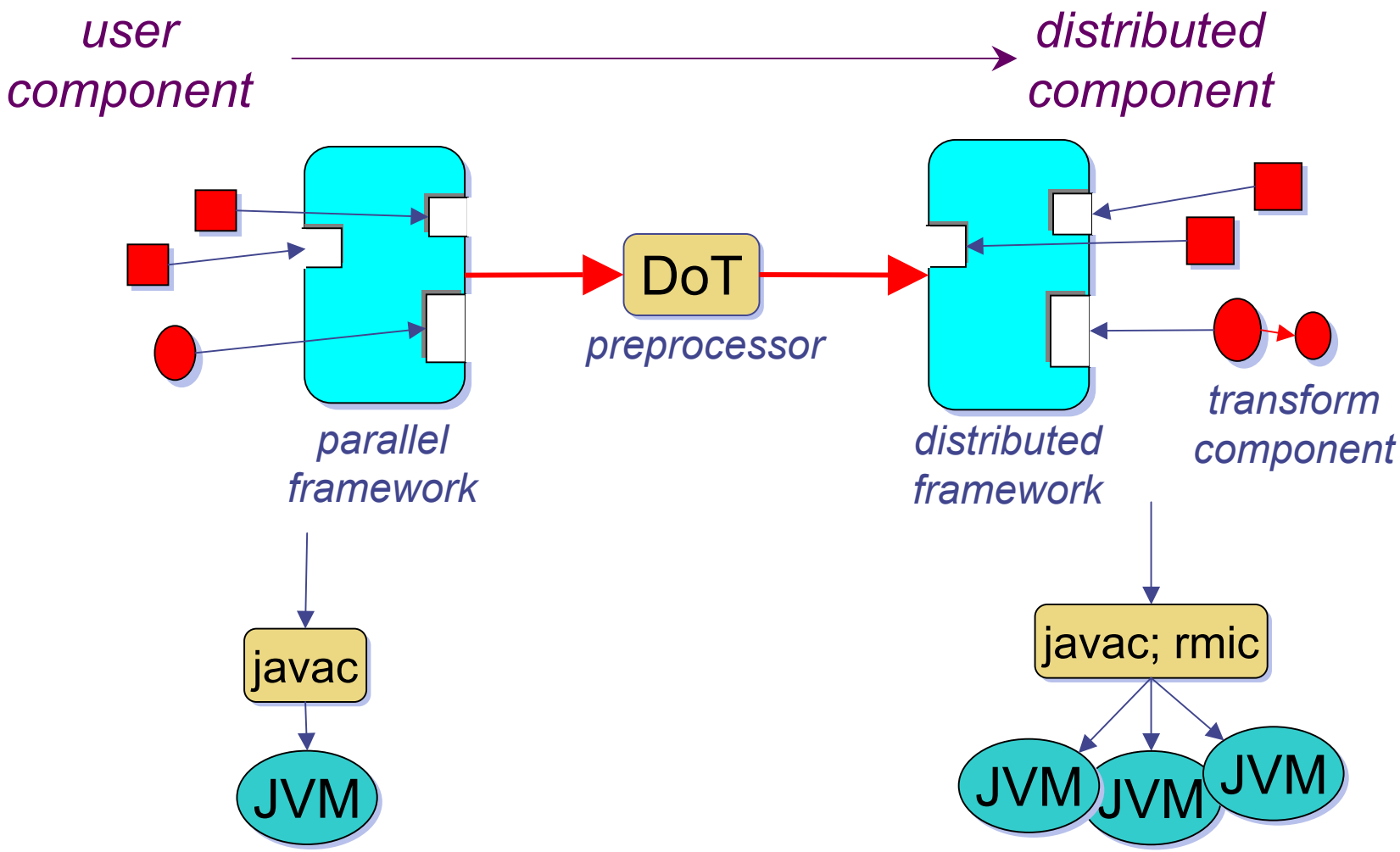
Programming distributed applications

- Provide a transparent distribution
  - *simplify or hide remote memory access*
- Structure parallel programming
  - *languages or extensions*
- Improve software quality
  - *Components, objects languages*

Main ideas of Do!

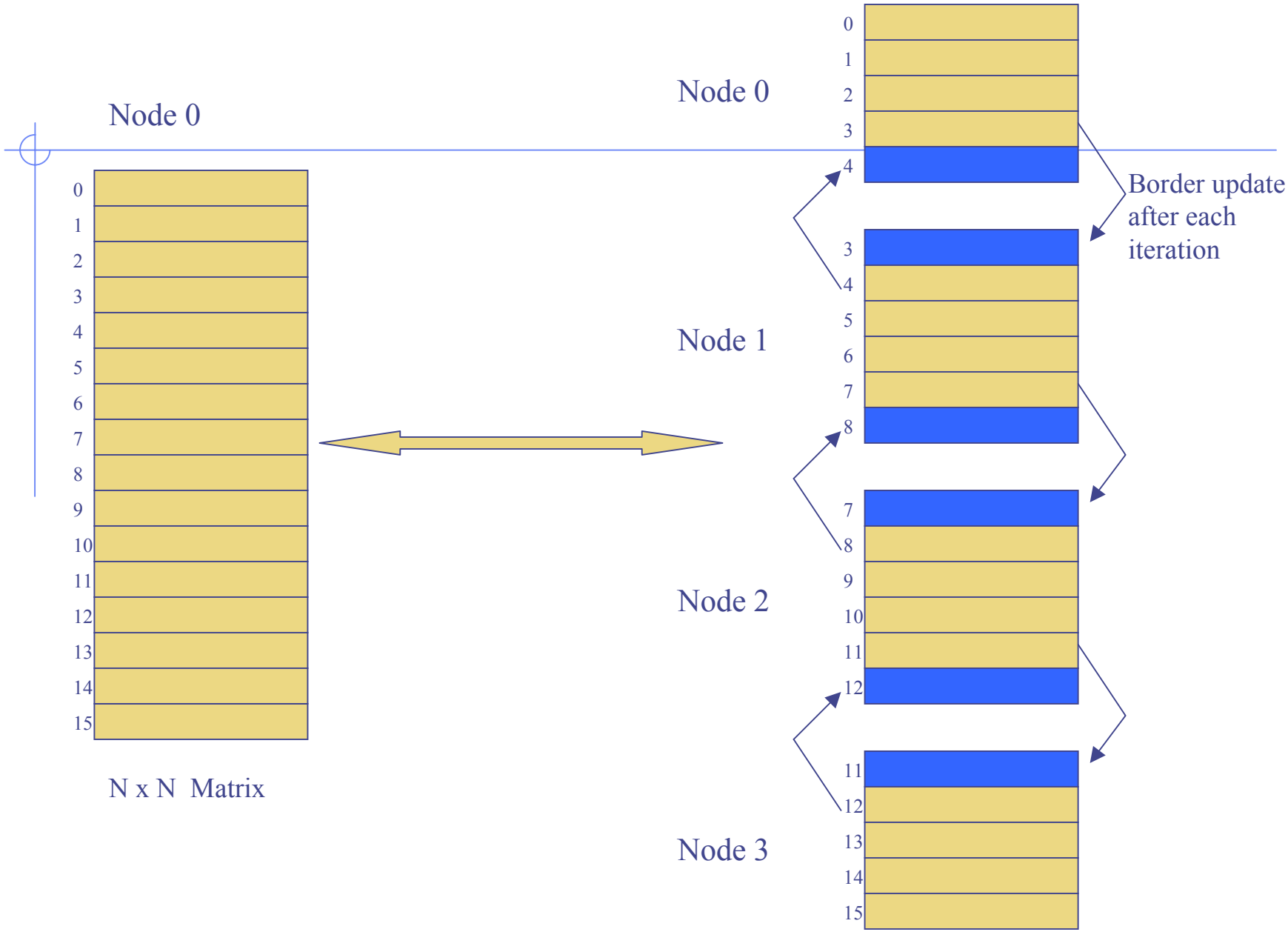
- 100% Java
- Parallelism is easier to describe than distribution
  - *write parallel, run distributed*
- Parallelism and distribution : distinct but tightly coupled
  - *integration of parallelism and distribution*
  - *distribution granularity different from parallelism granularity*

Do! environment



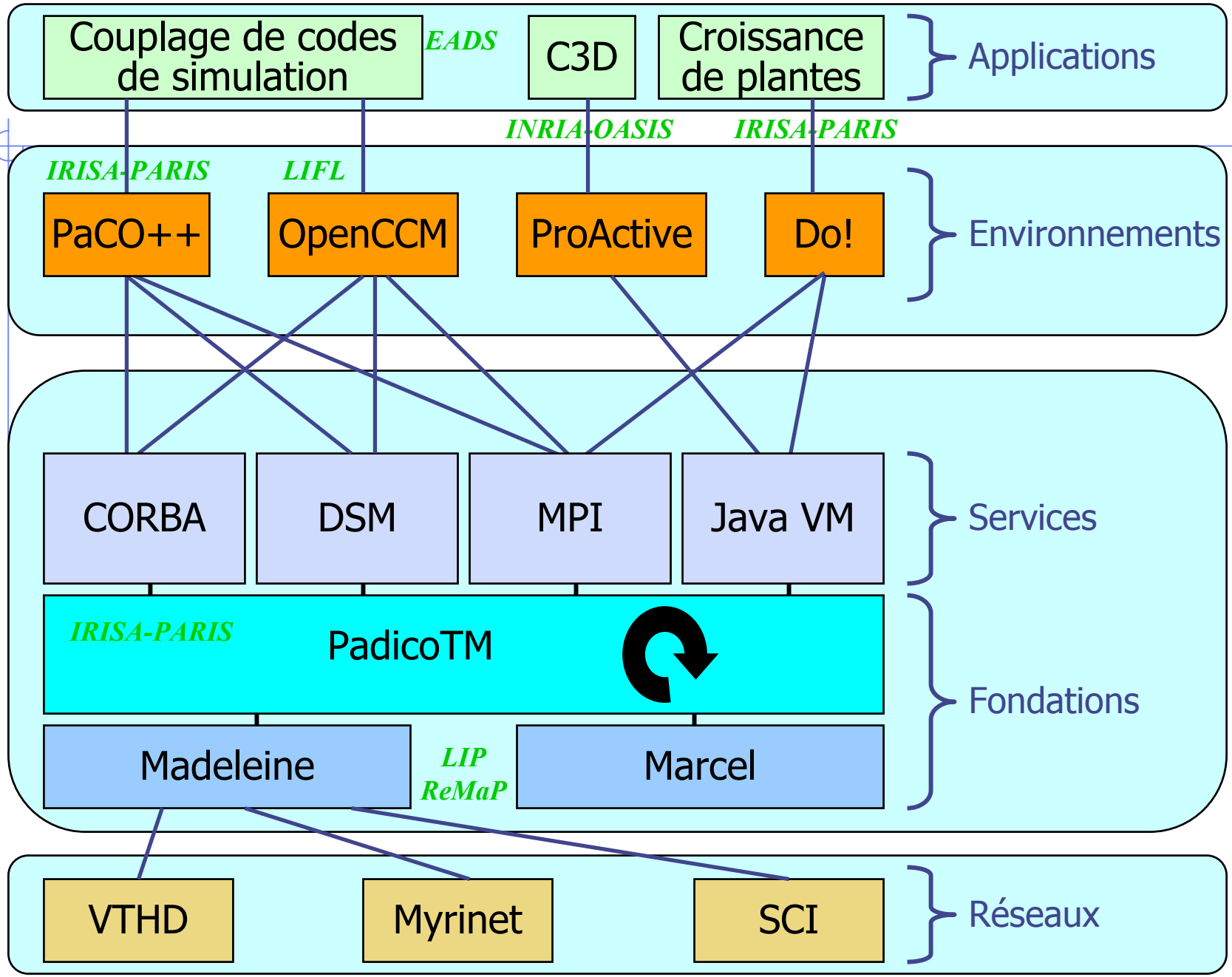
Extending Do! : distributed array

- Optimized support for “base type” collections
- Manage distributed array with overlap
  - Support for consistency management
  - Communication by messages (MPI)
- Currently
  - Not included in the translator
  - use RMI for messages


Node 0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
N x N Matrix
Node 0
0
1
2
3
4
Node 1
3
4
5
6
7
8
Node 2
7
8
9
10
11
12
Node 3
11
12
13
14
15
Border update after each iteration

Do ! and GRID-RMI

- Remove limitations
  - Do! has no deployment support
  - Do ! use standard JAVA RMI for messages
- Evaluate an advance runtime
  - Costs, advantages, drawback of the platform
  - Test of both MPI + RMI
- Experiments with an application: tree growth

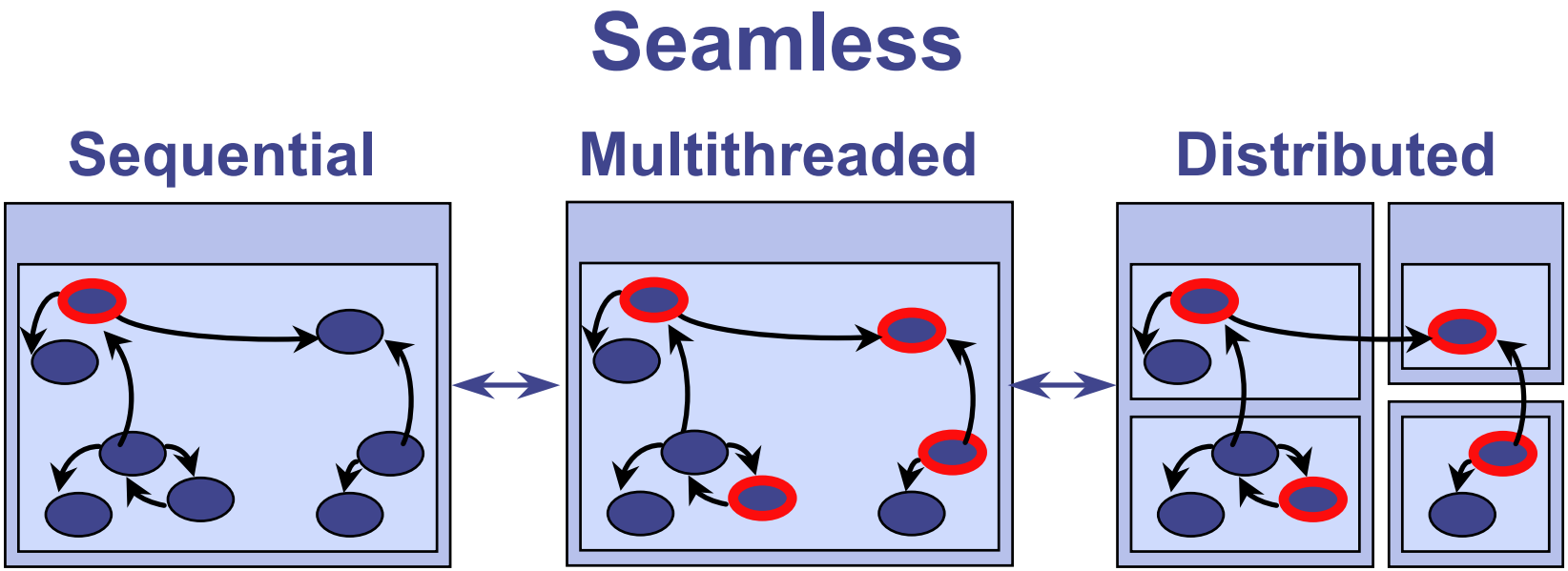

Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

ProActive

Denis Caromel & Françoise Baude
INRIA -- CNRS - I3S –
Univ. of Nice Sophia-Antipolis -- IUF

1. ProActive: Basic features
2. Deployment of ProActive applications
3. ProActive & GRID-RMI

1. ProActive :Objectives and Rationale

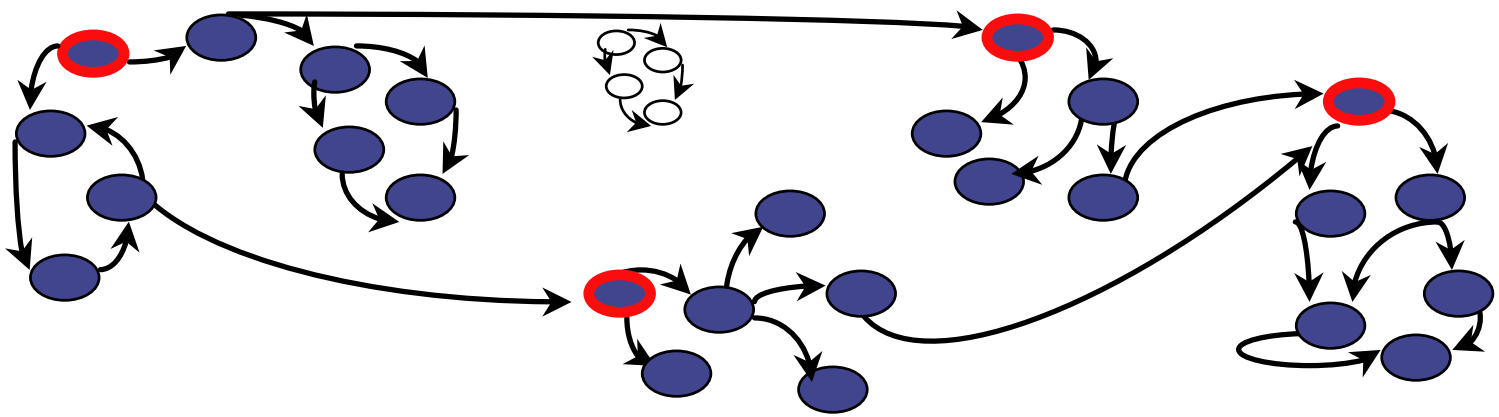


- Most of the time, activities and distribution are not known at the beginning, and change over time
- Seamless implies reuse, smooth and incremental transitions
- 100% Java

Library !

ProActive : model

- Active objects : coarse-grained structuring entities (subsystems)
- Each active object: - possibly owns many passive objects
  - has exactly one thread.
- No shared passive objects -- Parameters are passed by deep-copy
- Asynchronous Communication between active objects
- Future objects and wait-by-necessity.
- Full control to serve incoming requests (reification)



2. Deployment of ProActive applications (1)

- with XML-based descriptors
  - inside the application: active objects are created onto virtual nodes

    => the mapping (XML file) is independent of the effective location of nodes
  - virtual nodes manageable from Java
  - creation mechanism hide system

    => compatible with rsh, Globus, LSF, etc

2. Deployment of ProActive applications (2)

- with a graphical and interactive tool

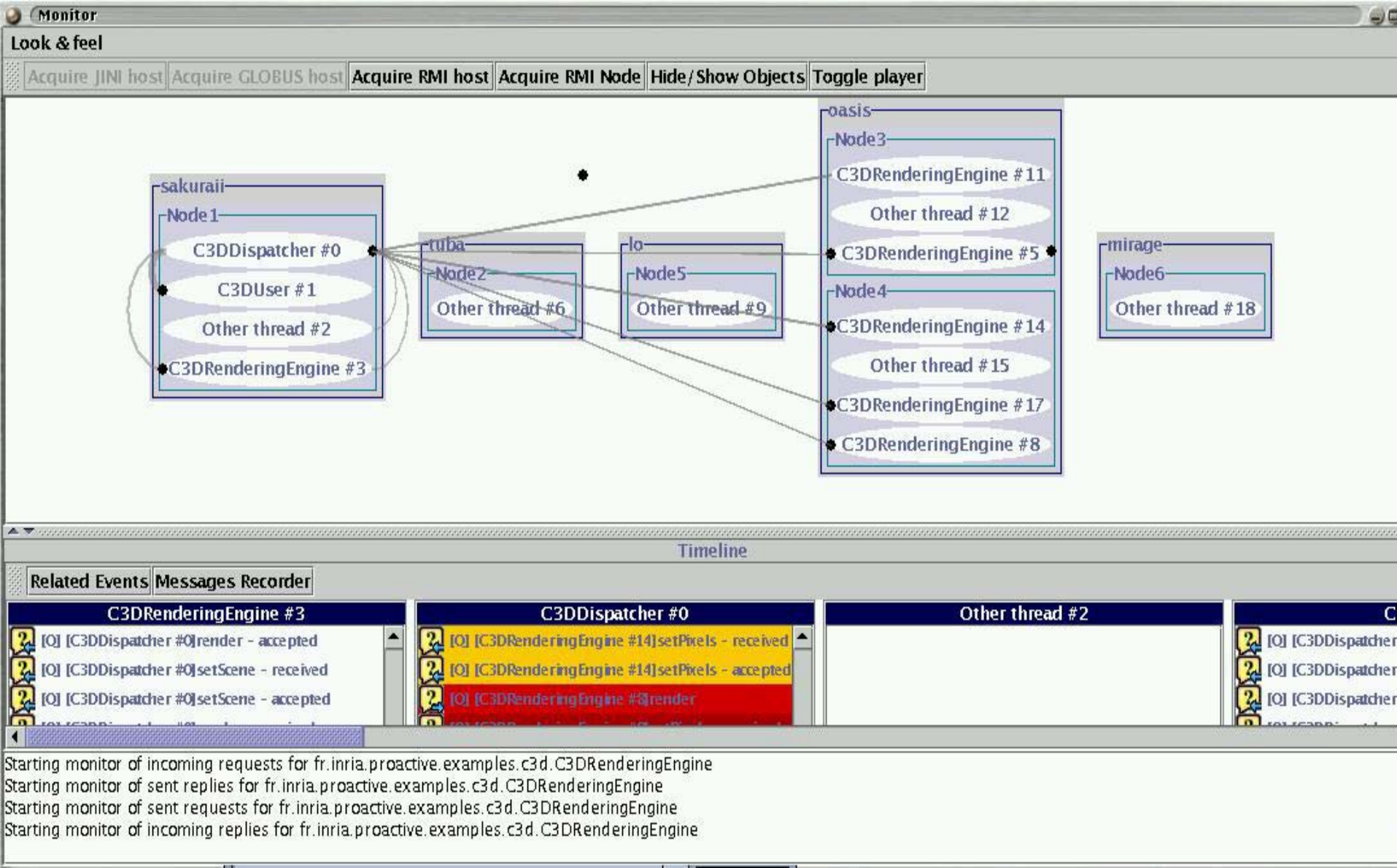
  IC2D: Interactive Control & Debug for Distribution

  work with any ProActive application

  Features:

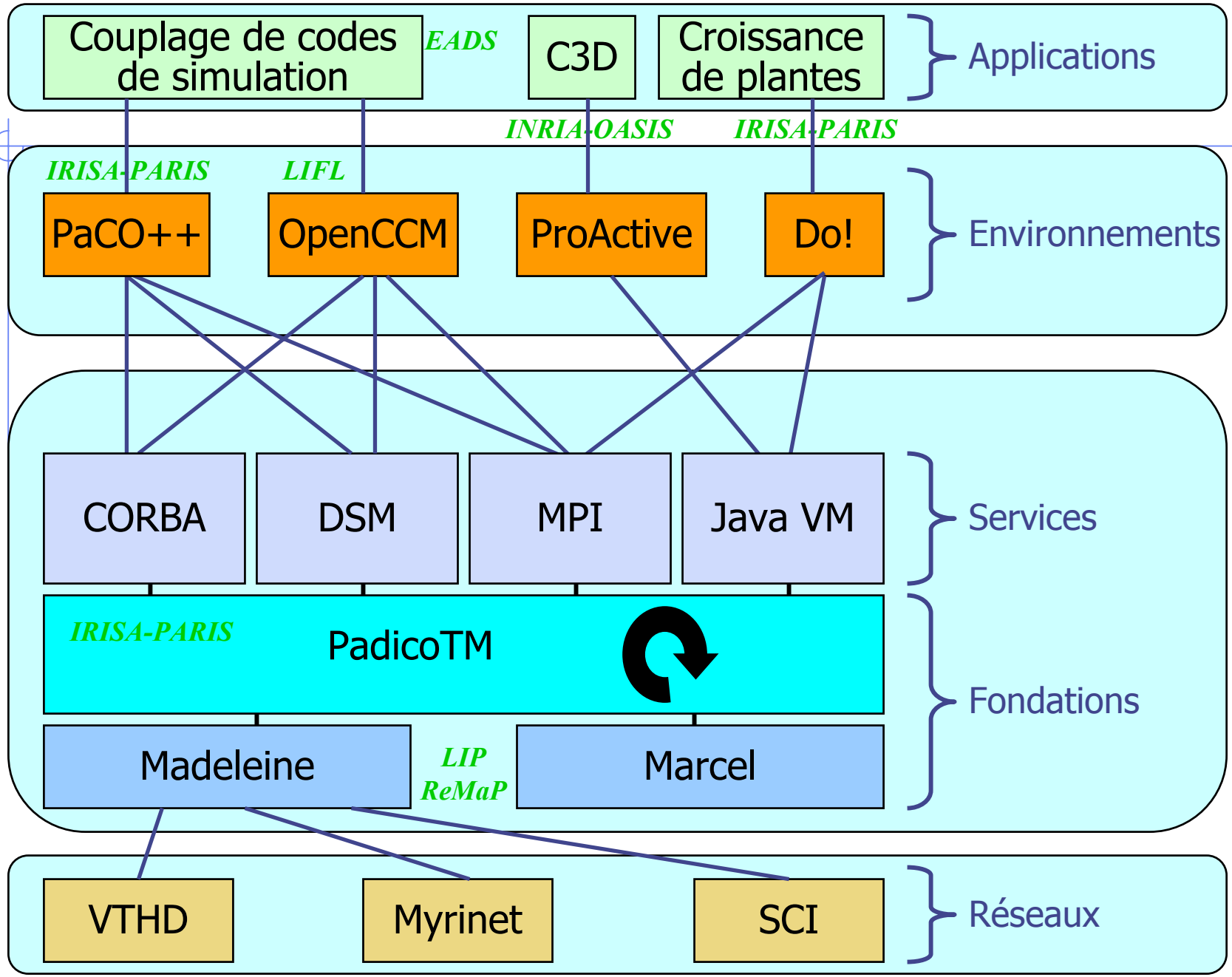
  1) Graphical and Textual visualization of hosts, VMs, nodes, active objects, communications
  2) Interactive Monitoring and Control of the mapping of active objects

IC2D on several machines



3. ProActive & GRID-RMI

- Port ProActive on a PadicoTM compatible JVM
  - hypothesis: efficient threads and sockets operations
  - Performance improvement: may reconsider part of ProActive
- Group communications within ProActive
  - goal: provide high level and efficient group communication primitives in Java
  - target: parallel codes running in hierarchical cluster.
  - problems: efficient implantation (slow RMI serialization)
  - extension of IC2D to group communications
- Test, deploy large applications in ProActive within a grid computing environment
  - Application type: C3D
  - Code coupling


Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

PaCO++

Portable Parallel CORBA Objet

Alexandre Denis Christian Pérez
Thierry Priol André Ribes

Projet PARIS

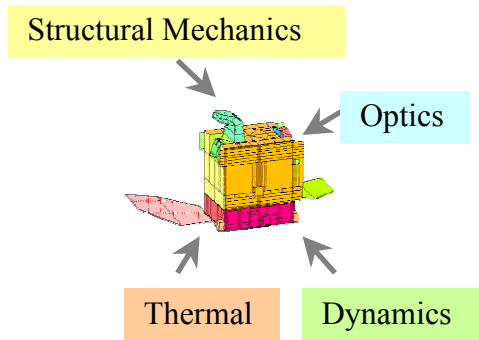
IRISA/INRIA, Rennes

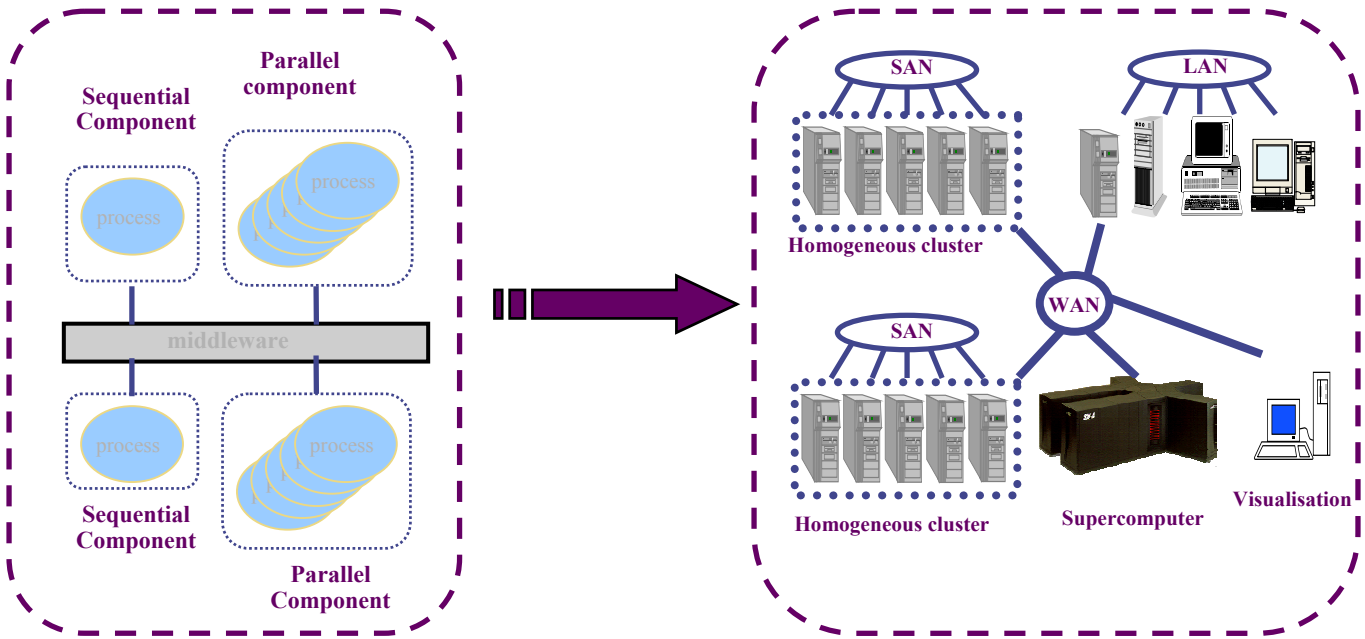
http://www.irisa.fr/paris/Padico/

Code coupling

- Structured application
  - Visible communication hierarchy
- Code coupling
  - Parallel codes encapsulated into objects
  - Objects connected through a middleware

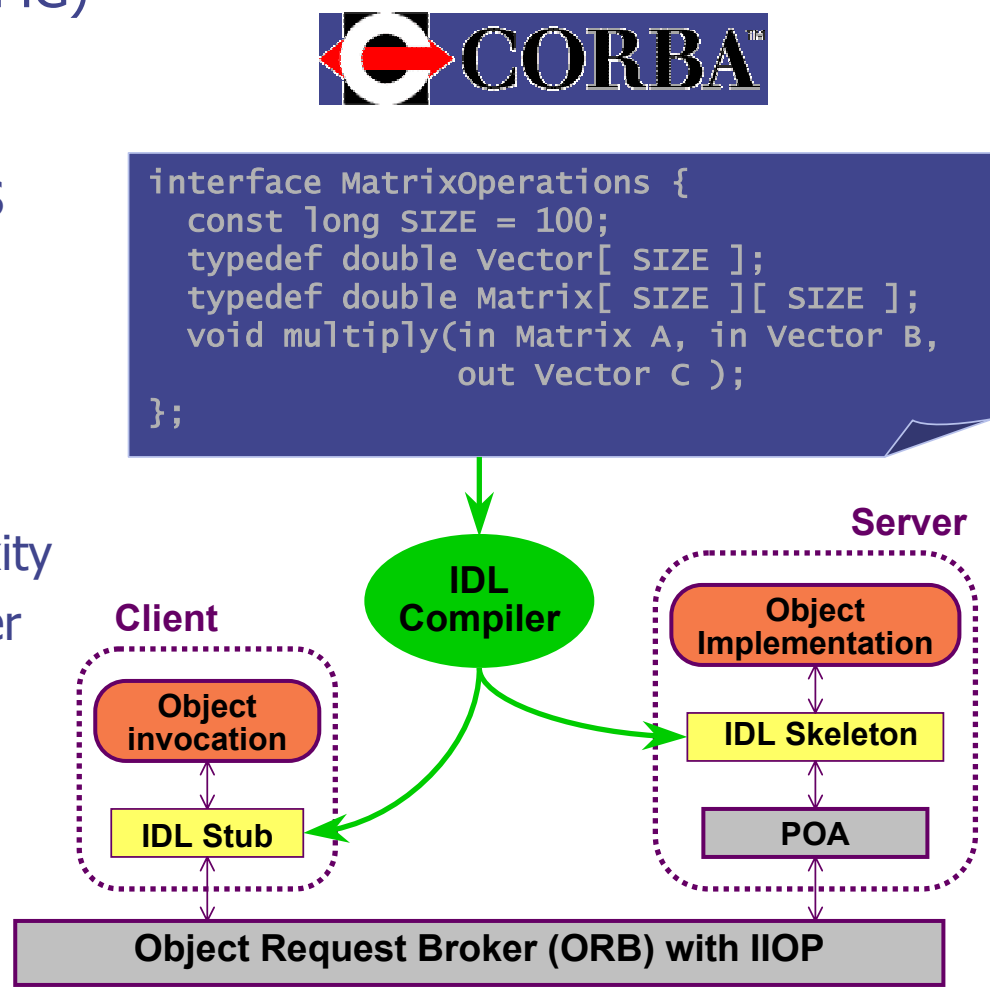
➔ *Need for parallel objects*





Overview of CORBA

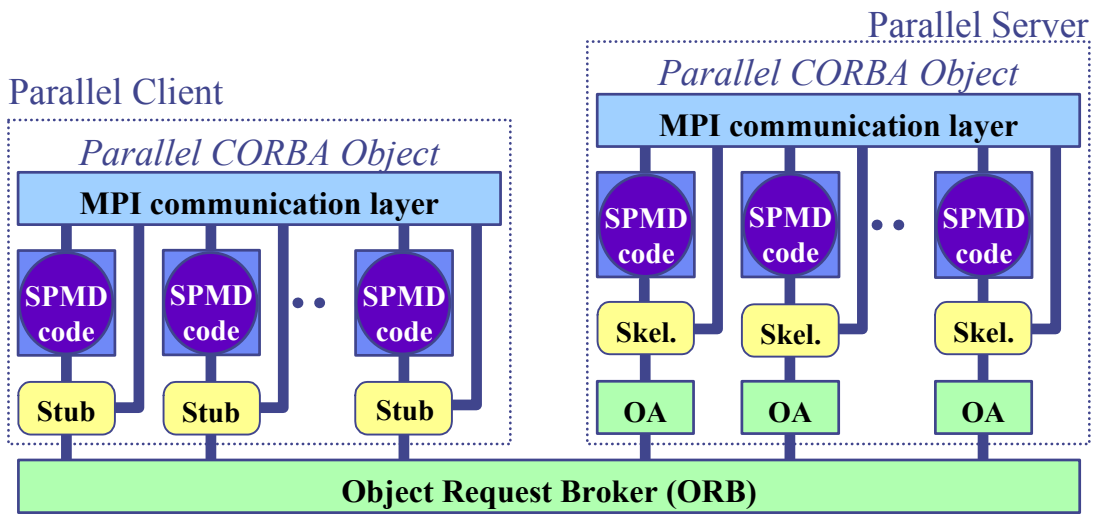
- Distributed Object concept (OMG)
  - Object oriented programming
  - Remote method invocation
  - Independence: Hardware, OS and programming language
  - Vendor independence (interoperability)
- Advantages
  - Reduce the software complexity
  - Well suited for control transfer
- Drawback
  - Lack of support for parallel programming



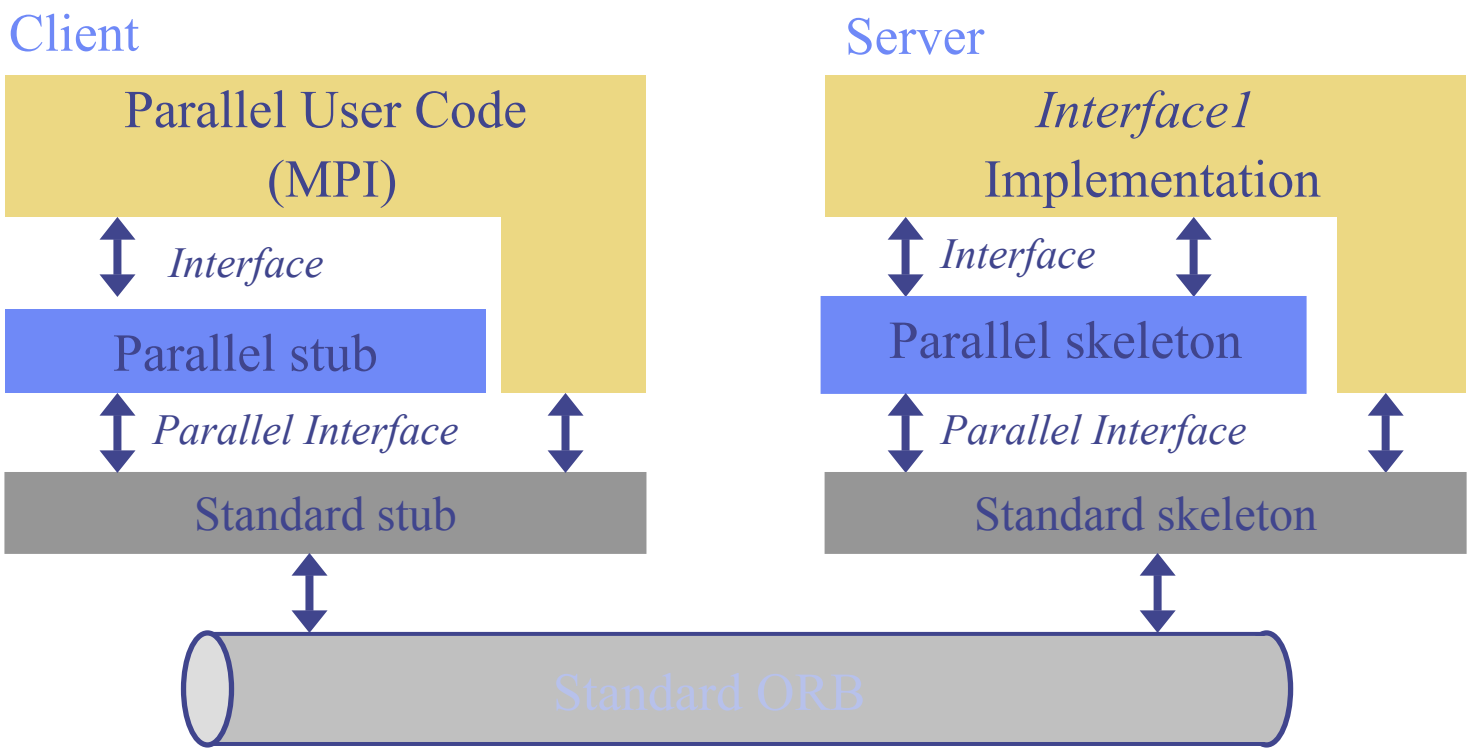
Portable Parallel CORBA Objects

- Objectives
  - No modification of CORBA standard
  - Look like regular CORBA objects
  - Data distribution transparency
  - Interoperable with non-parallel aware ORB

```
I_var o = I::_narrow(...);
o->fun();
```



Software architecture

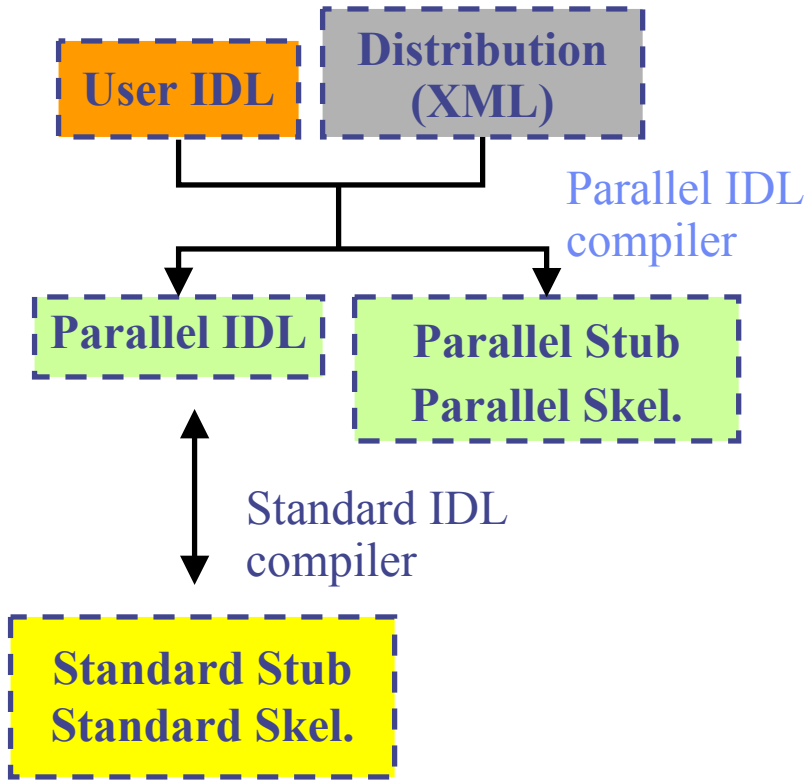


Parallelism management

- XML file: parallelism description
  - Method: scalar, parallel, …
  - Argument: replicated, distributed
    - "distributed" IDL type

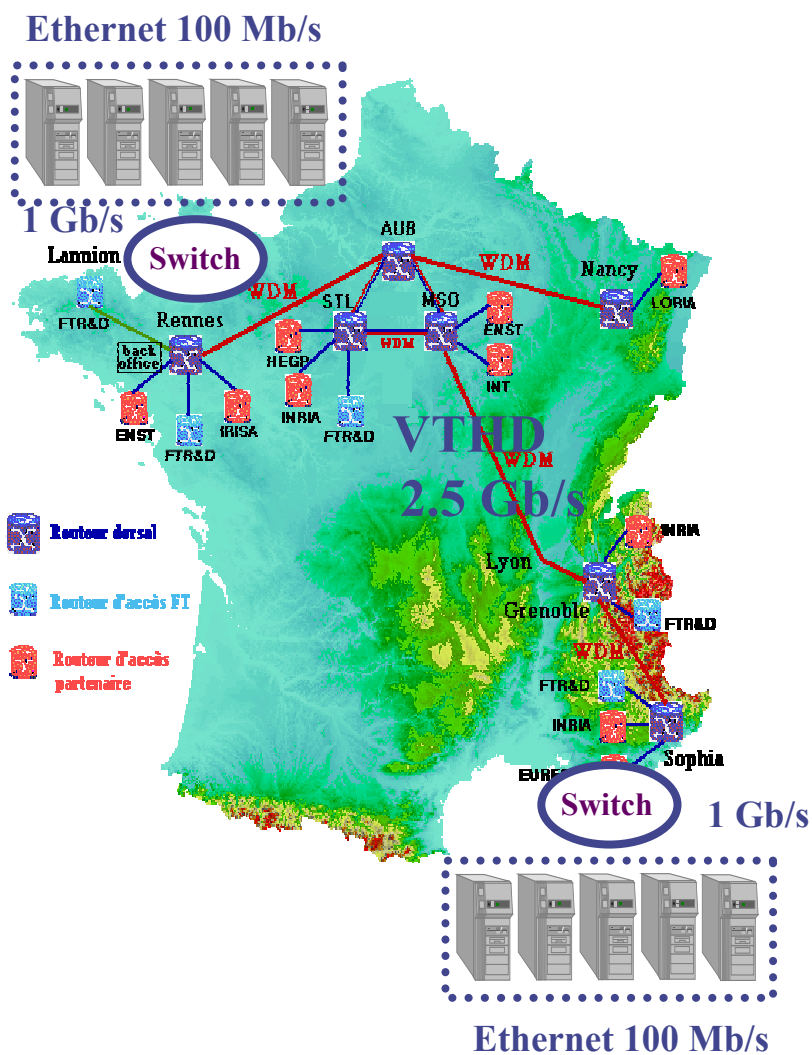
```
Interface IExample
 Method_name: send_data
 Method_type: parallel
 Return_type: none
 Number_argument: 1
    Type_Argument1: Distributed
```

- Modification of the IDL file
  - methods renaming
  - support of distributed types



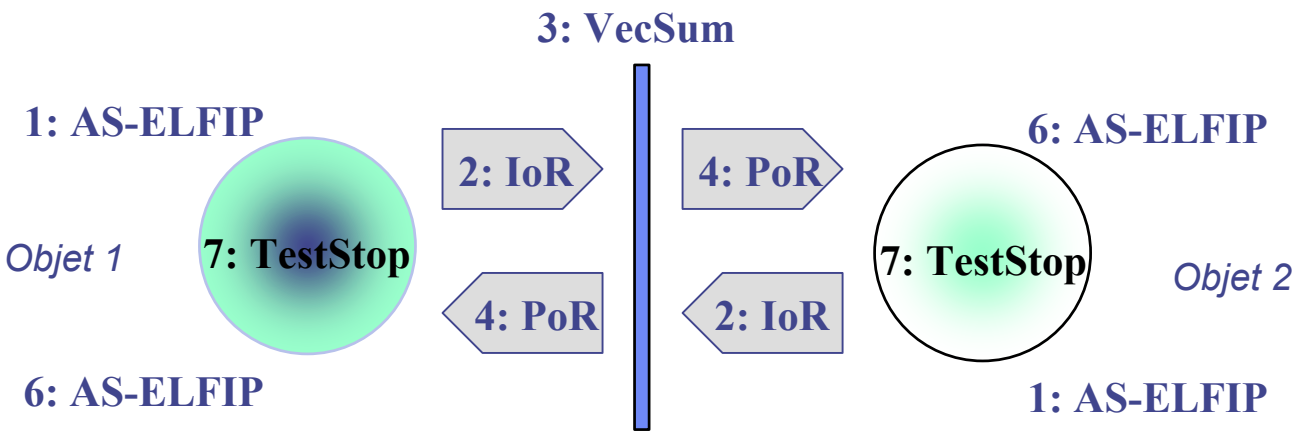
PaCO++ performance

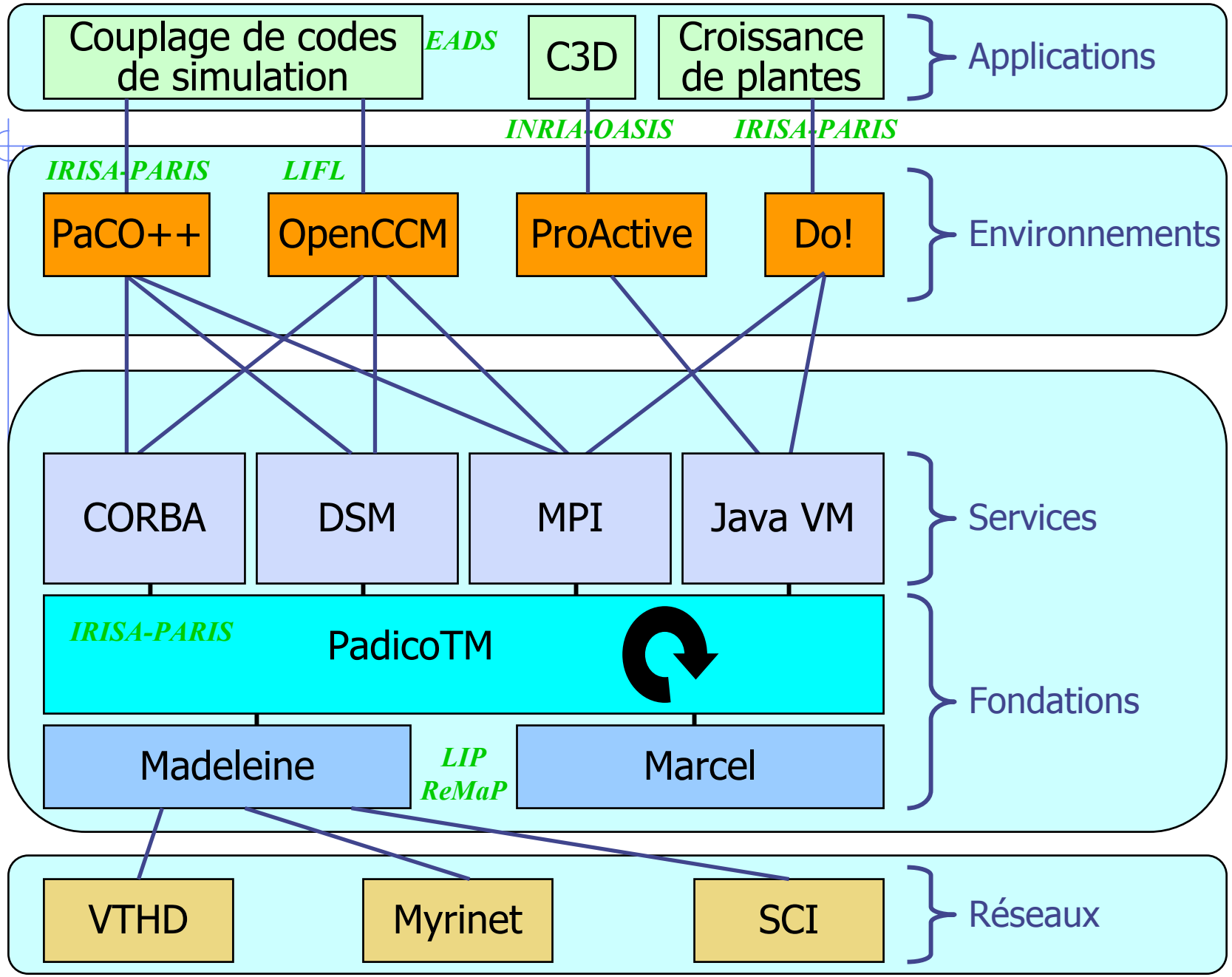
- Client and server programs
  - PaCO++
  - OmniORB (AT&T)
  - MPI
- WAN Network
  - VTHD (2.5 Gb/s)
- Performances
  - 11 nodes to 11 nodes
  - 826 Mb/s (103 MB/s)
    - ⇒ at the 1 Gbit/s switch limit
  - Pt-2-Pt at 75 Mb/s (9.4 MB/s)



PaCO++ & GRID-RMI

- Develop specifications
- Alpha version of PaCO++ compiler
- Targeted ORB: omniORB & MICO
- Validation with EADS application
  - Original: sequential file transfer between MPI codes



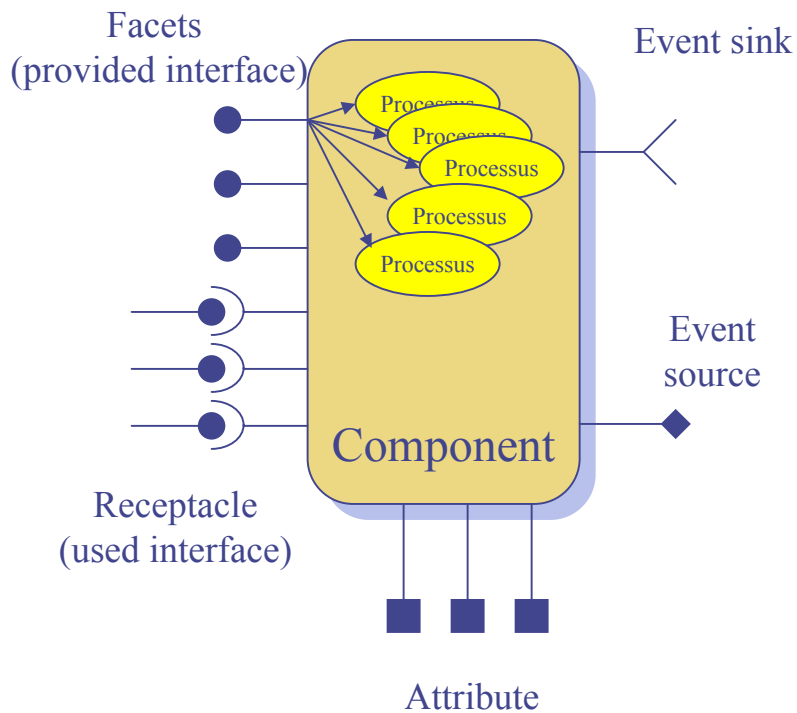

Couplage de codes de simulation
EADS
C3D
Croissance de plantes
Applications
INRIA-OASIS
IRISA-PARIS
IRISA-PARIS
LIFL
PaCO++
OpenCCM
ProActive
Do!
Environnements
CORBA
DSM
MPI
Java VM
Services
IRISA-PARIS
PadicoTM
Madeleine
LIP
ReMaP
Marcel
Fondations
VTHD
Myrinet
SCI
Réseaux

OpenCCM

Philippe.Merle@lifl.fr
équipe GOAL - LIFL – USTL
INRIA

Software component

- Software component
  - Code re-use
  - Modularity
  - Independent unit of deployment
- Application = assembly of components

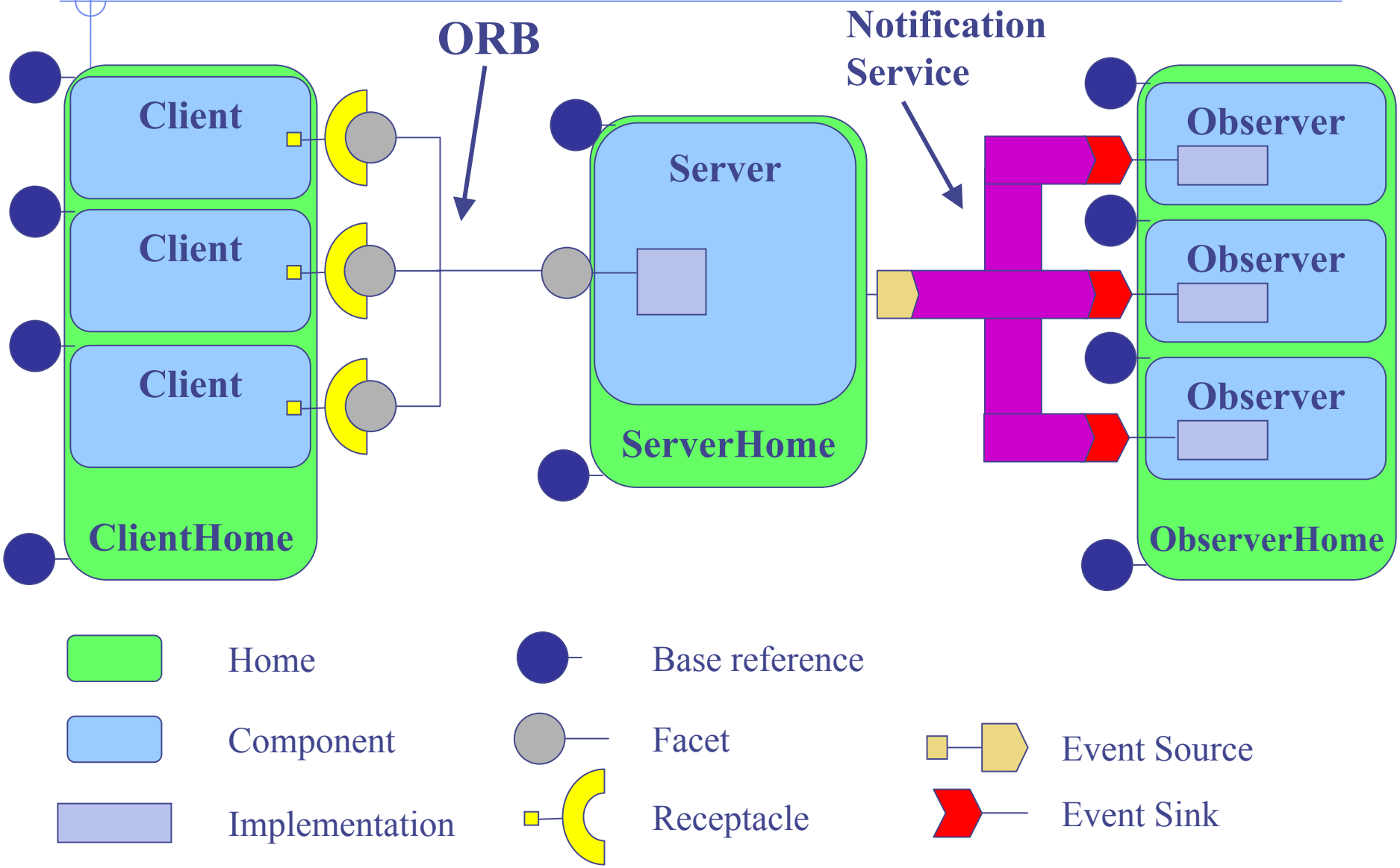


CORBA component

CORBA 3.0 Component model

- Open OMG Standard OMG for distributed components
  - multi-languages, multi-OSs, multi-ORBs, multi-venders, etc.
- A component model with ports
- Distributed packaging and deployment technology
- Containers to manage life cycle, activation, security, transactions, persistence and events
- Compatibility with Enterprise Java Beans

Distributed application with CCM



OpenCCM - Goals

- To be the reference implementation of CCM
  - Support the all CCM model
- Provide an open implementation
  - Multi-ORBs and multi-langages (Java, C++, IDLscript)
  - Extensile and adaptable to do CCM++
- Provide an experimentation framework for R&D
  - Open container to integrate new services
  - Reliable deployment at large scale
  - Component trader
- Open software distributed by ObjectWeb consortium
  - *http://www.objectweb.org/OpenCCM/index.html*

ACI GRID RMI & OpenCCM

- Modern programmation approach
  - ***Experiment distributed components for grid applications***
- High performance
  - ***Extend OpenCCM for C++ components***
  - ***Generate C++ code and C++ runtime***
- Deployment at large scale (~ thousands of node)
  - ***Evaluate OpenCCM and extend its deployment capabilities***
- Integration of new services, e.g. parallel components
  - ***Evaluate and extend the open OpenCCM container***

Conclusion

- Goals of GRID-RMI
  - Promote a programming model for the Grid that combine both parallel and distributed programming
  - Hide network heterogeneity
  - Provide high performance
- Runtime components: Marcel-Madeleine/Padico
- Programming model: Do!, ProActive, PaCO++, OpenCCM
- **`http://www.irisa.fr/Grid-RMI`**