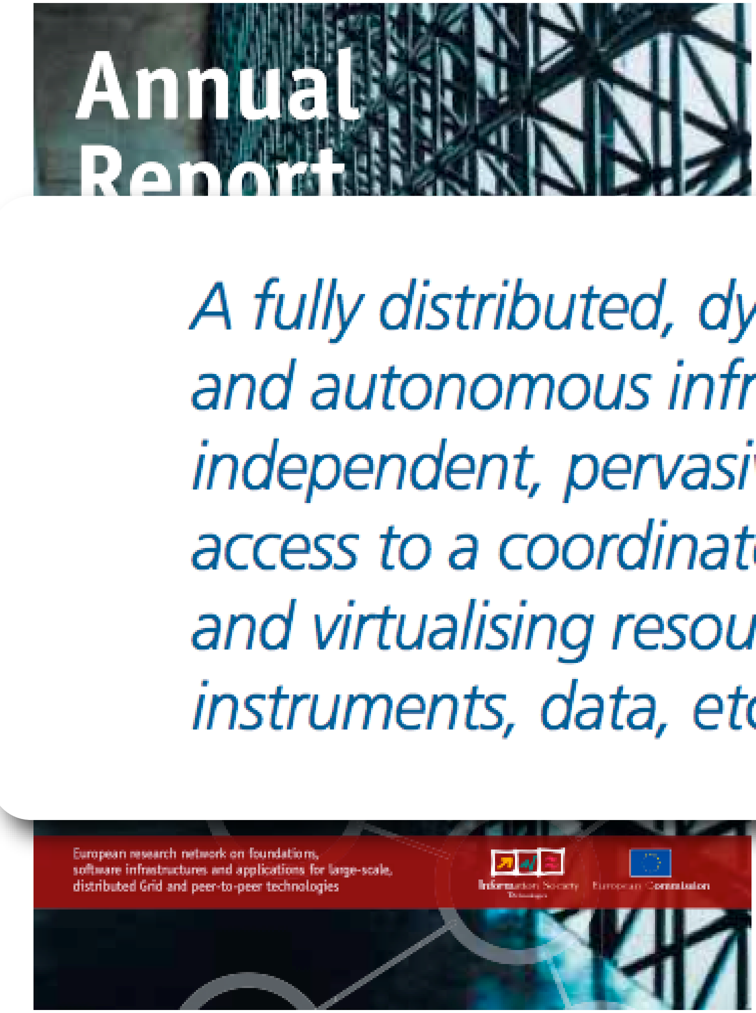
Bridging Global Computing with Grid (BIGG) Programming model section

Marco Danelutto
Dept. Computer Science - Univ. Pisa &
CoreGRID Programming model Institute

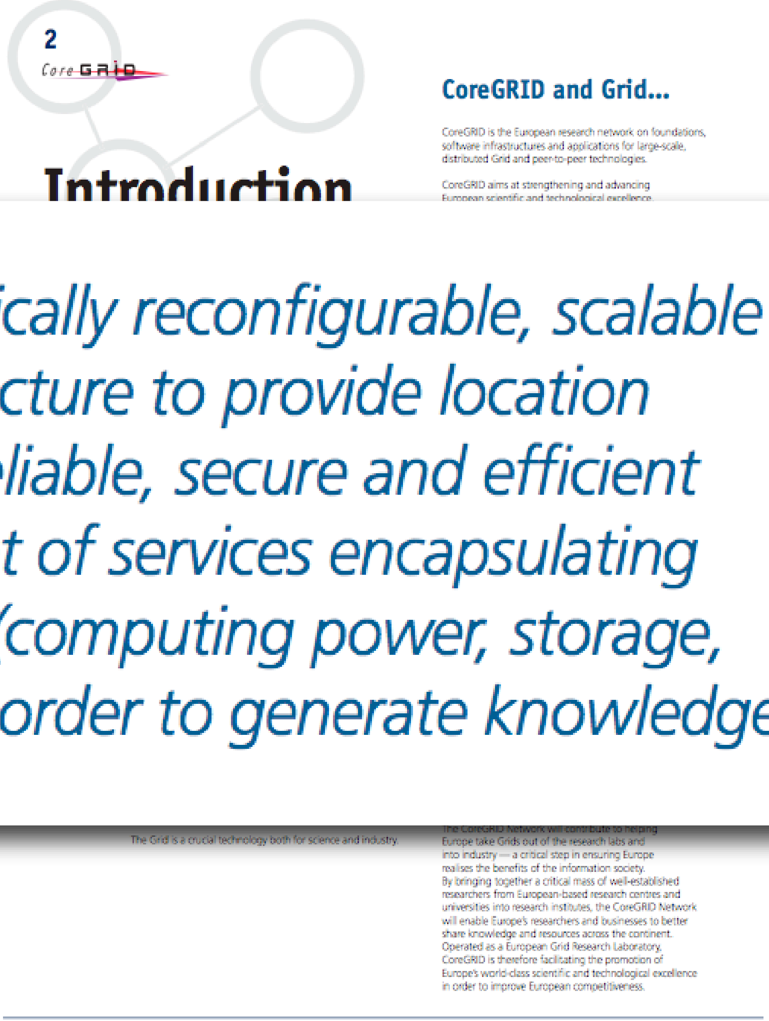
Sophia Antipolis, November 29, 2006

GRID definition



Annual Report

European research network on foundations, software infrastructures and applications for large-scale, distributed Grid and peer-to-peer technologies



2

Introduction

CoreGRID and Grid...

CoreGRID is the European research network on foundations, software infrastructures and applications for large-scale, distributed Grid and peer-to-peer technologies.

CoreGRID aims at strengthening and advancing European scientific and technological excellence.

The Grid is a crucial technology both for science and industry.

The CoreGRID Network will contribute to helping Europe take Grids out of the research labs and into industry — a critical step in ensuring Europe realises the benefits of the information society. By bringing together a critical mass of well-established researchers from European-based research centres and universities into research institutes, the CoreGRID Network will enable Europe's researchers and businesses to better share knowledge and resources across the continent. Operated as a European Grid Research Laboratory, CoreGRID is therefore facilitating the promotion of Europe's world-class scientific and technological excellence in order to improve European competitiveness.

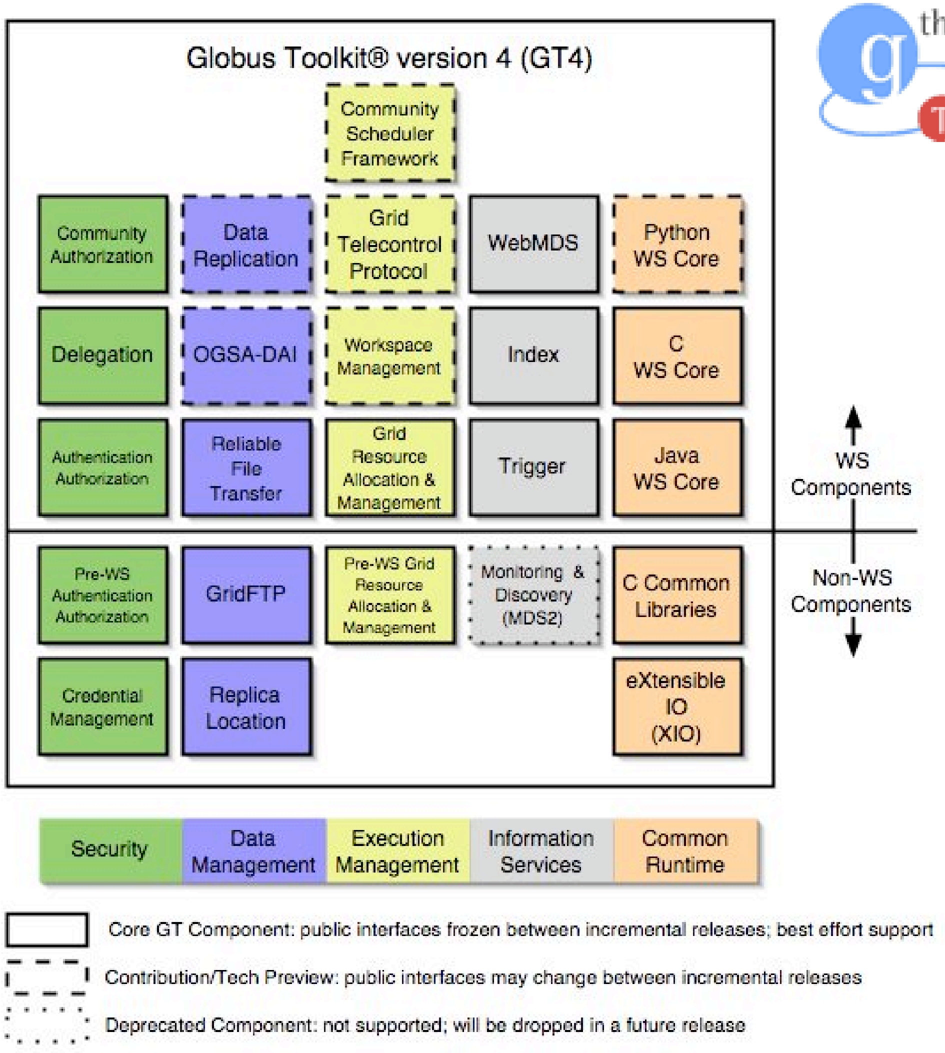
A fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualising resources (computing power, storage, instruments, data, etc.) in order to generate knowledge.

Programmer perspective (today)

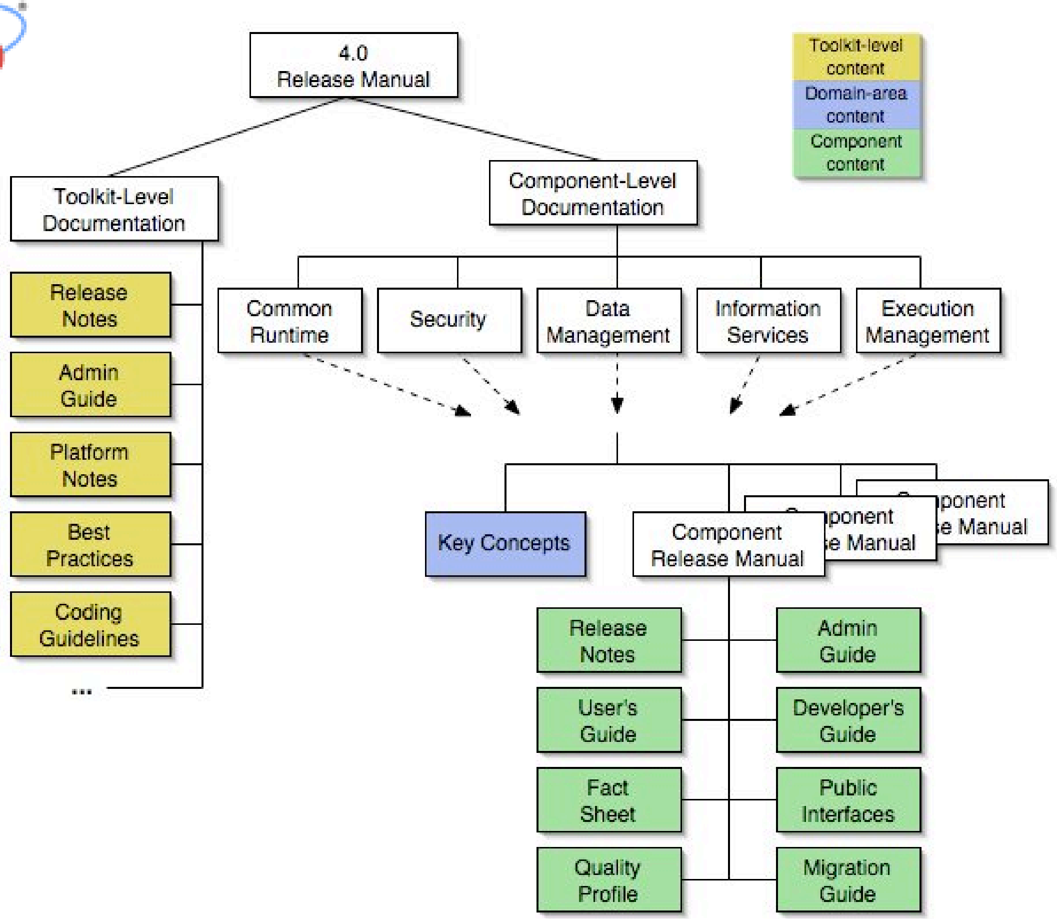
- *collection of heterogeneous resources, dynamically available in time, subject to faults & misfunctioning, searched, recruited, secured, deployed and used to perform complex tasks*

- concurrent activity set up, mapping and scheduling; resource management (recruiting, monitoring, repairing); communication & synchronization management; fault tolerance and security management; heterogeneity management (cross compiling, architectural neutral data formats), ...

Programmer perspective (today) (2)

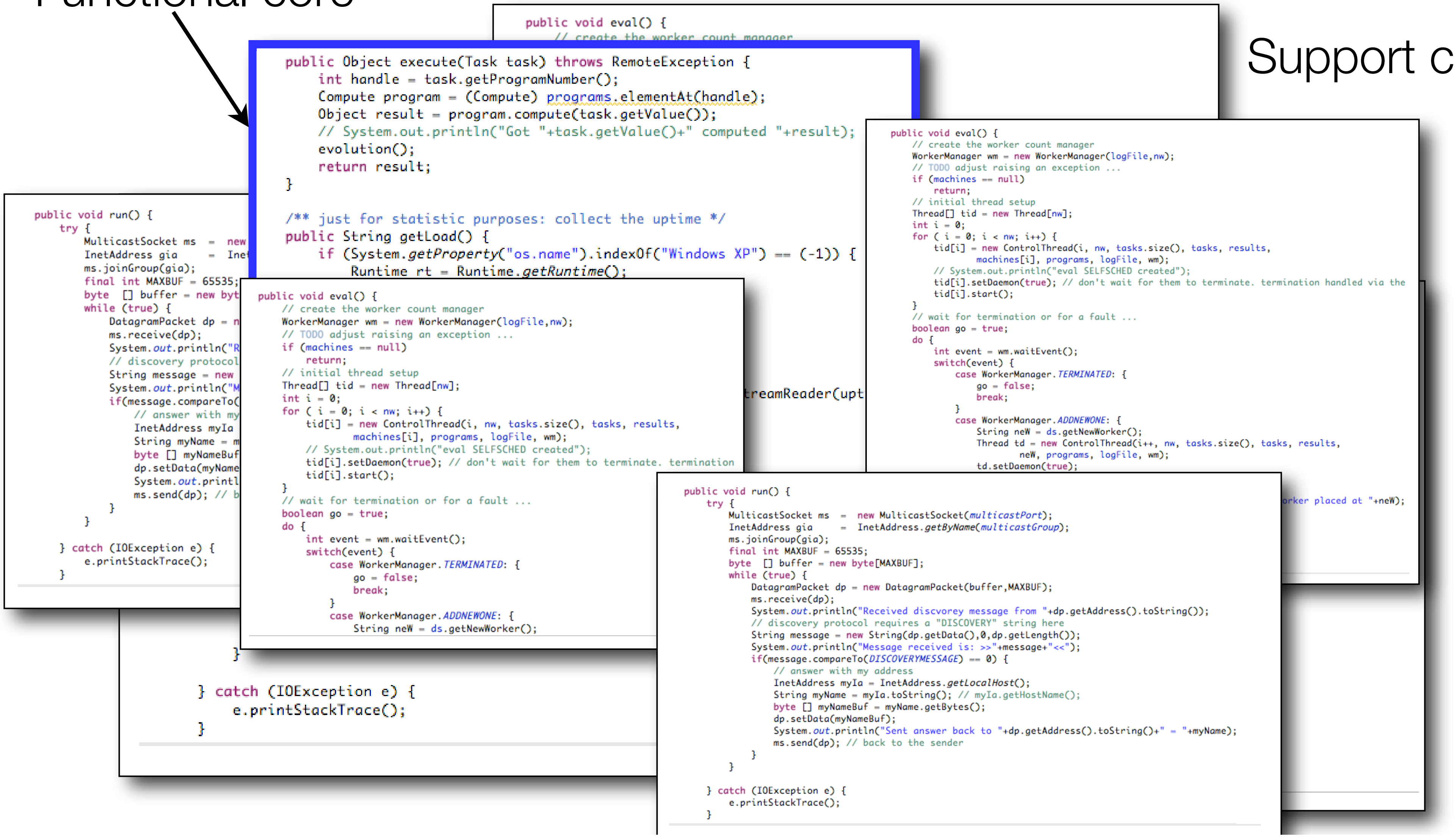

Globus Toolkit® version 4 (GT4)
Community Scheduler Framework
Community Authorization
Data Replication
Grid Telecontrol Protocol
WebMDS
Python WS Core
Delegation
OGSA-DAI
Workspace Management
Index
C WS Core
Authentication Authorization
Reliable File Transfer
Grid Resource Allocation & Management
Trigger
Java WS Core
WS Components
Non-WS Components
Pre-WS Authentication Authorization
GridFTP
Pre-WS Grid Resource Allocation & Management
Monitoring & Discovery (MDS2)
C Common Libraries
Credential Management
Replica Location
eXtensible IO (XIO)
Security
Data Management
Execution Management
Information Services
Common Runtime
Core GT Component: public interfaces frozen between incremental releases; best effort support
Contribution/Tech Preview: public interfaces may change between incremental releases
Deprecated Component: not supported; will be dropped in a future release

the globus® toolkit



Programmer perspective (today) (3)

Functional core



```
public Object execute(Task task) throws RemoteException {
    int handle = task.getProgramNumber();
    Compute program = (Compute) programs.elementAt(handle);
    Object result = program.compute(task.getValue());
    // System.out.println("Got "+task.getValue()+" computed "+result);
    evolution();
    return result;
}

/** just for statistic purposes: collect the uptime */
public String getLoad() {
    if (System.getProperty("os.name").indexOf("Windows XP") == (-1)) {
        Runtime rt = Runtime.getRuntime();
```

```
public void eval() {
    // create the worker count manager
```

```
treamReader(upt
```

```
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
public void eval() {
    // create the worker count manager
    WorkerManager wm = new WorkerManager(logFile,nw);
    // TODO adjust raising an exception ...
    if (machines == null)
        return;
    // initial thread setup
    Thread[] tid = new Thread[nw];
    int i = 0;
    for ( i = 0; i < nw; i++) {
        tid[i] = new ControlThread(i, nw, tasks.size(), tasks, results,
                machines[i], programs, logFile, wm);
        // System.out.println("eval SELFSCHED created");
        tid[i].setDaemon(true); // don't wait for them to terminate. termination handled via the
        tid[i].start();
    }
    // wait for termination or for a fault ...
    boolean go = true;
    do {
        int event = wm.waitEvent();
        switch(event) {
            case WorkerManager.TERMINATED: {
                go = false;
                break;
            }
            case WorkerManager.ADDNEWONE: {
                String neW = ds.getNewWorker();
                Thread td = new ControlThread(i++, nw, tasks.size(), tasks, results,
                        neW, programs, logFile, wm);
                td.setDaemon(true);
```

```
orker placed at "+neW);
```

```
public void run() {
    try {
        MulticastSocket ms  =  new MulticastSocket(multicastPort);
        InetAddress gia     =  InetAddress.getByName(multicastGroup);
        ms.joinGroup(gia);
        final int MAXBUF = 65535;
        byte  [] buffer = new byte[MAXBUF];
        while (true) {
            DatagramPacket dp = new DatagramPacket(buffer,MAXBUF);
            ms.receive(dp);
            System.out.println("Received discvorey message from "+dp.getAddress().toString());
            // discovery protocol requires a "DISCOVERY" string here
            String message = new String(dp.getData(),0,dp.getLength());
            System.out.println("Message received is: >>"+message+"<<");
            if(message.compareTo(DISCOVERYMESSAGE) == 0) {
                // answer with my address
                InetAddress myIa = InetAddress.getLocalHost();
                String myName = myIa.toString(); // myIa.getHostName();
                byte [] myNameBuf = myName.getBytes();
                dp.setData(myNameBuf);
                System.out.println("Sent answer back to "+dp.getAddress().toString()+" = "+myName);
                ms.send(dp); // back to the sender
            }
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
```

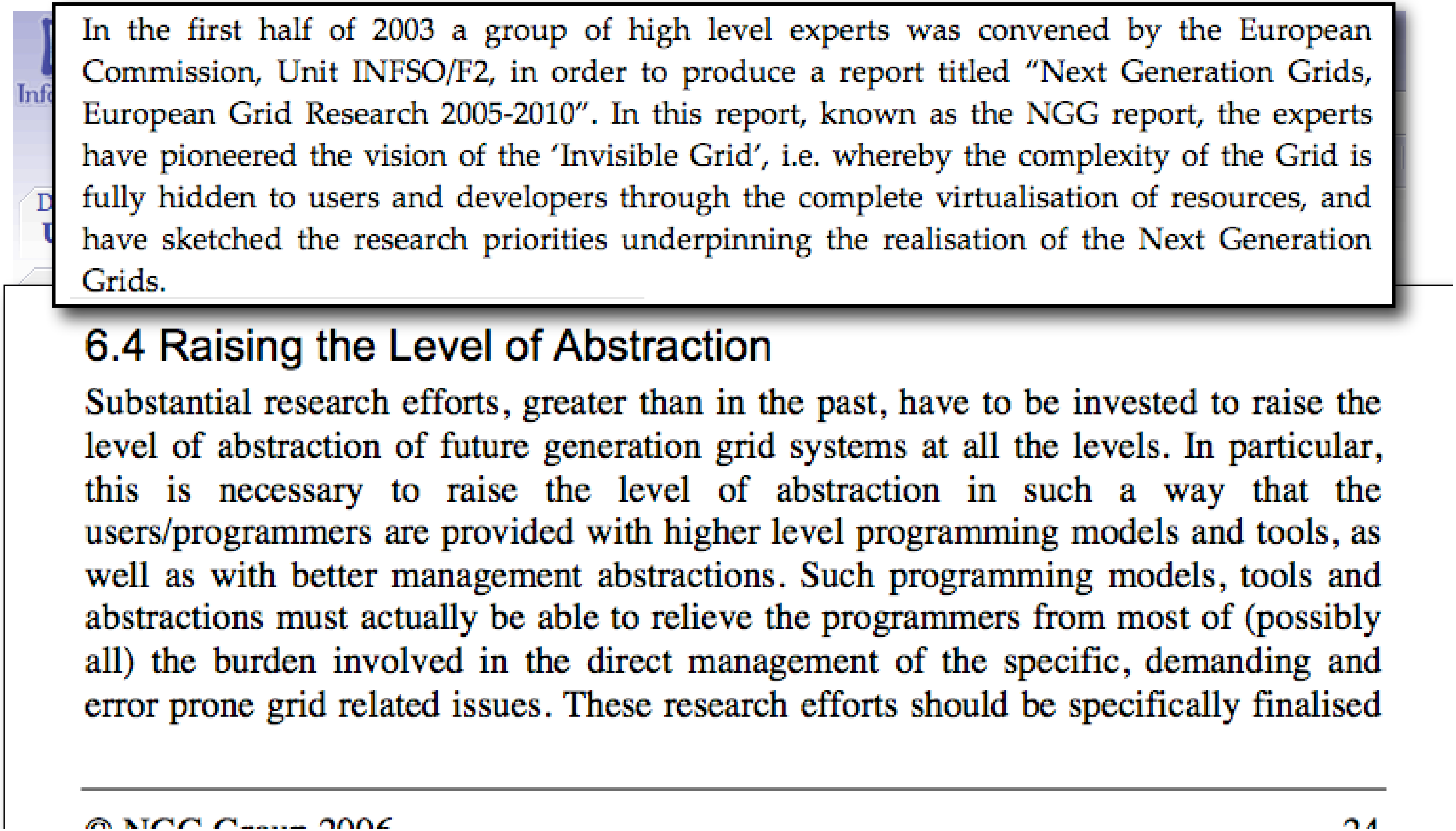
```
public void run() {
    try {
        MulticastSocket ms  =  new
        InetAddress gia     =  Inet
        ms.joinGroup(gia);
        final int MAXBUF = 65535;
        byte  [] buffer = new byt
        while (true) {
            DatagramPacket dp = n
            ms.receive(dp);
            System.out.println("R
            // discovery protocol
            String message = new
            System.out.println("M
            if(message.compareTo(
                // answer with my
                InetAddress myIa
                String myName = m
                byte [] myNameBuf
                dp.setData(myName
                System.out.printl
                ms.send(dp); // b
            }
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
public void eval() {
    // create the worker count manager
    WorkerManager wm = new WorkerManager(logFile,nw);
    // TODO adjust raising an exception ...
    if (machines == null)
        return;
    // initial thread setup
    Thread[] tid = new Thread[nw];
    int i = 0;
    for ( i = 0; i < nw; i++) {
        tid[i] = new ControlThread(i, nw, tasks.size(), tasks, results,
                machines[i], programs, logFile, wm);
        // System.out.println("eval SELFSCHED created");
        tid[i].setDaemon(true); // don't wait for them to terminate. termination
        tid[i].start();
    }
    // wait for termination or for a fault ...
    boolean go = true;
    do {
        int event = wm.waitEvent();
        switch(event) {
            case WorkerManager.TERMINATED: {
                go = false;
                break;
            }
            case WorkerManager.ADDNEWONE: {
                String neW = ds.getNewWorker();
```

Support code

NGG recommendations



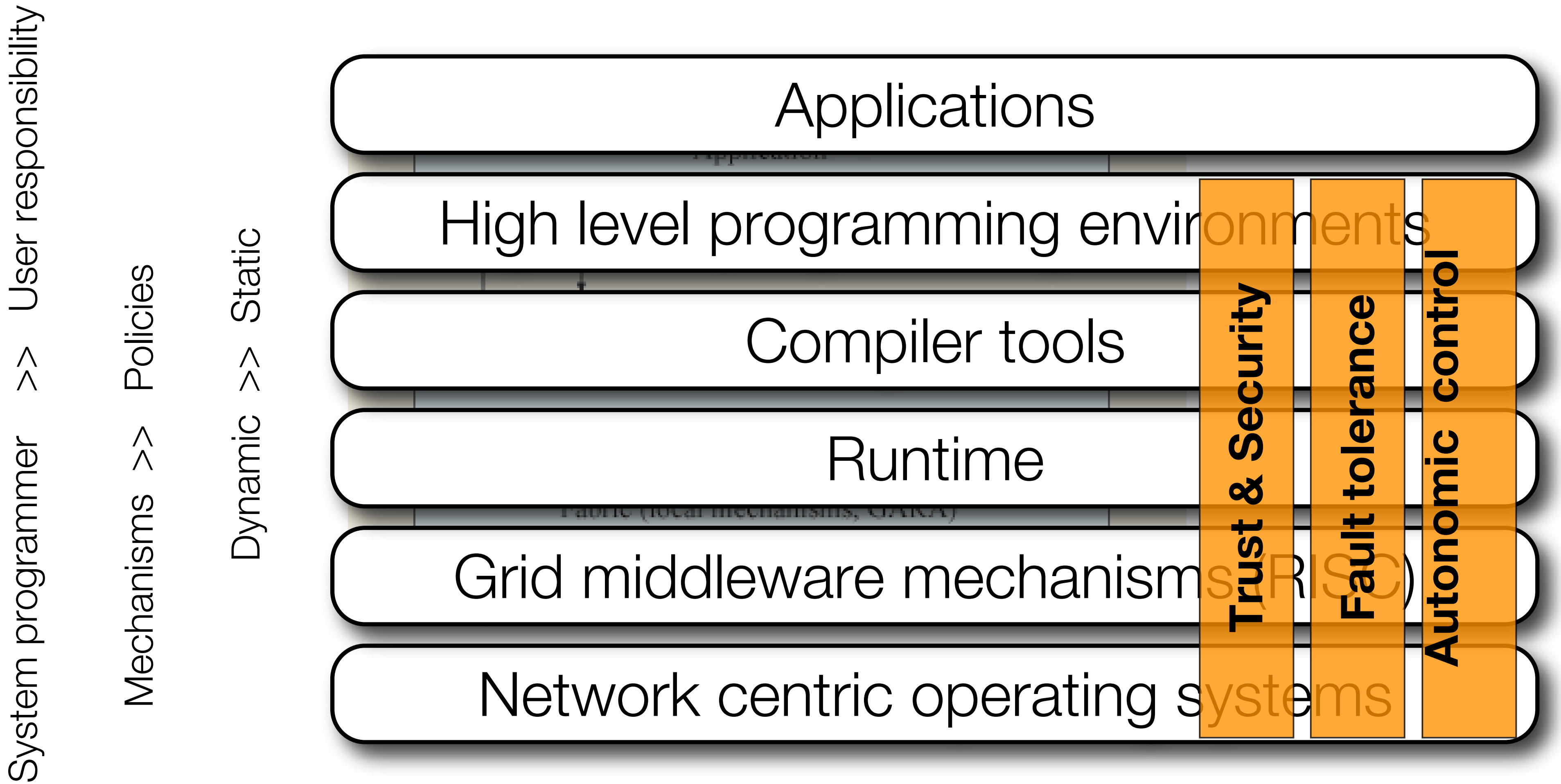
In the first half of 2003 a group of high level experts was convened by the European Commission, Unit INFSO/F2, in order to produce a report titled "Next Generation Grids, European Grid Research 2005-2010". In this report, known as the NGG report, the experts have pioneered the vision of the 'Invisible Grid', i.e. whereby the complexity of the Grid is fully hidden to users and developers through the complete virtualisation of resources, and have sketched the research priorities underpinning the realisation of the Next Generation Grids.

6.4 Raising the Level of Abstraction

Substantial research efforts, greater than in the past, have to be invested to raise the level of abstraction of future generation grid systems at all the levels. In particular, this is necessary to raise the level of abstraction in such a way that the users/programmers are provided with higher level programming models and tools, as well as with better management abstractions. Such programming models, tools and abstractions must actually be able to relieve the programmers from most of (possibly all) the burden involved in the direct management of the specific, demanding and error prone grid related issues. These research efforts should be specifically finalised

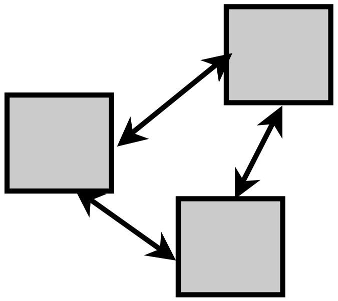
How to implement NGG recommendation ?

- Exploit *layered* design of grid middleware + programming environments

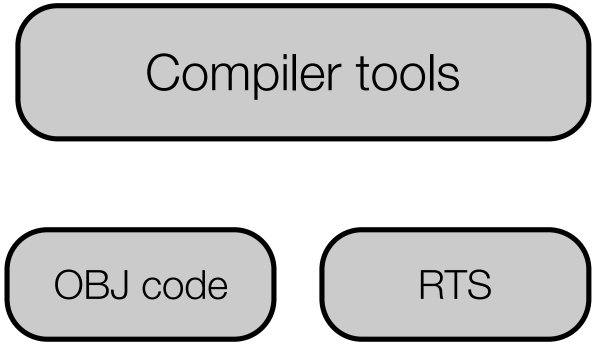

System programmer >> User responsibility
Mechanisms >> Policies
Dynamic >> Static
Applications
High level programming environments
Compiler tools
Runtime
Grid middleware mechanisms (RISC)
Network centric operating systems
Trust & Security
Fault tolerance
Autonomic control

Plus ...

- Advanced programming models
  - providing predefined, customizable, efficient programming skeletons - design/coordinations patterns



  - move away from assembly language
  - implemented by compilers or RTS/libraries (exploiting layer hierarchy)



Compiler tools

OBJ code

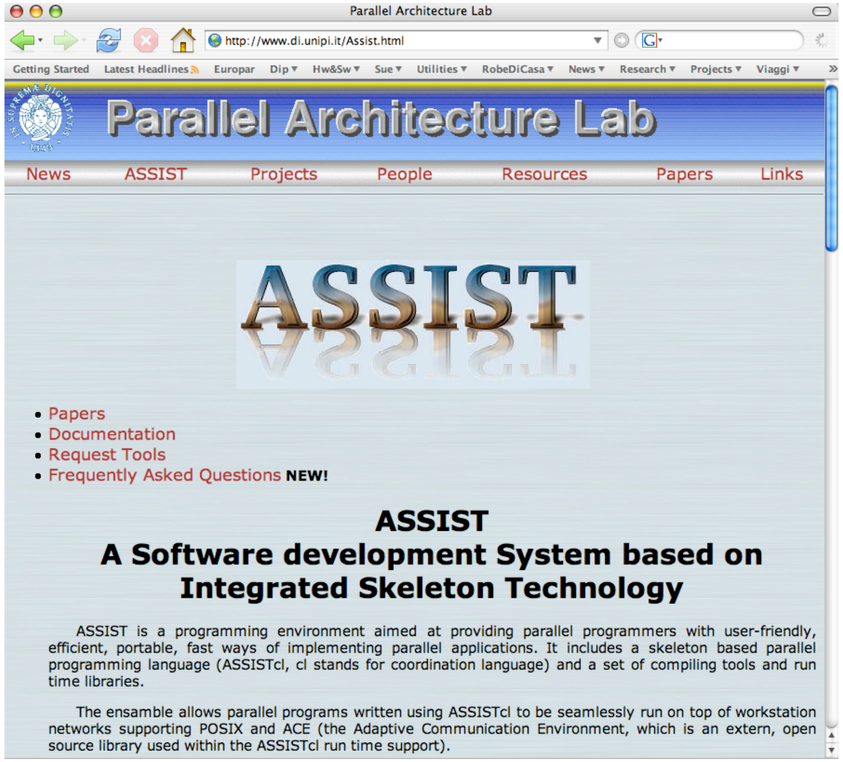
RTS

  - relieve programmers of unnecessary burden,
    - possibly free them !

Is it possible ?

grid.it
project

- ASSIST high level, structured, parallel programming environment
- compiler + run time + grid middleware
- heterogeneous node handling + adaptivity (performance contracts)
- up & running since 2001 (advanced features in 2005)



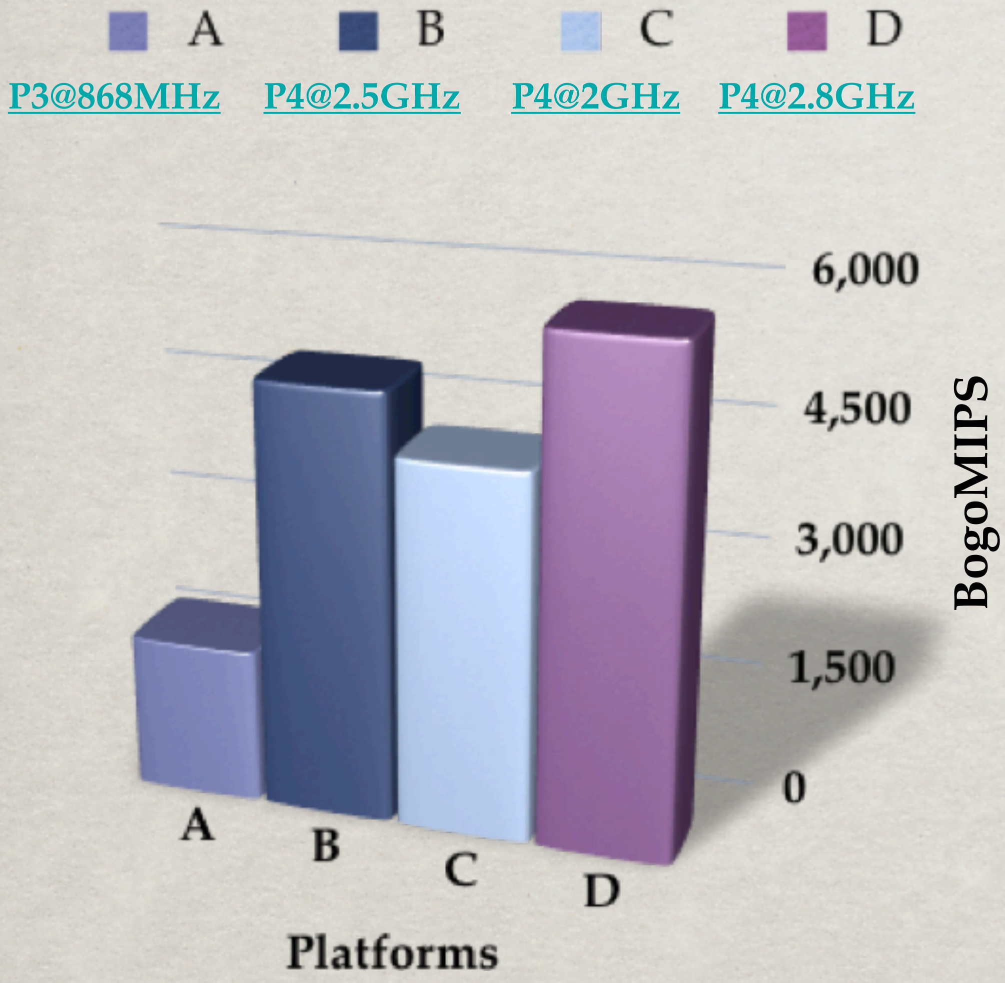
- Papers
- Documentation
- Request Tools
- Frequently Asked Questions **NEW!**

ASSIST
A Software development System based on Integrated Skeleton Technology

ASSIST is a programming environment aimed at providing parallel programmers with user-friendly, efficient, portable, fast ways of implementing parallel applications. It includes a skeleton based parallel programming language (ASSISTcl, cl stands for coordination language) and a set of compiling tools and run time libraries.

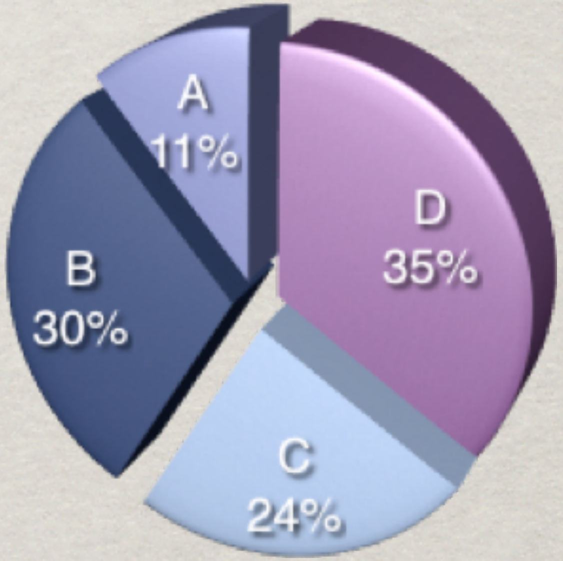
The ensamble allows parallel programs written using ASSISTcl to be seamlessly run on top of workstation networks supporting POSIX and ACE (the Adaptive Communication Environment, which is an extern, open source library used within the ASSISTcl run time support).

Exp 2: Data-Parallel(STP)

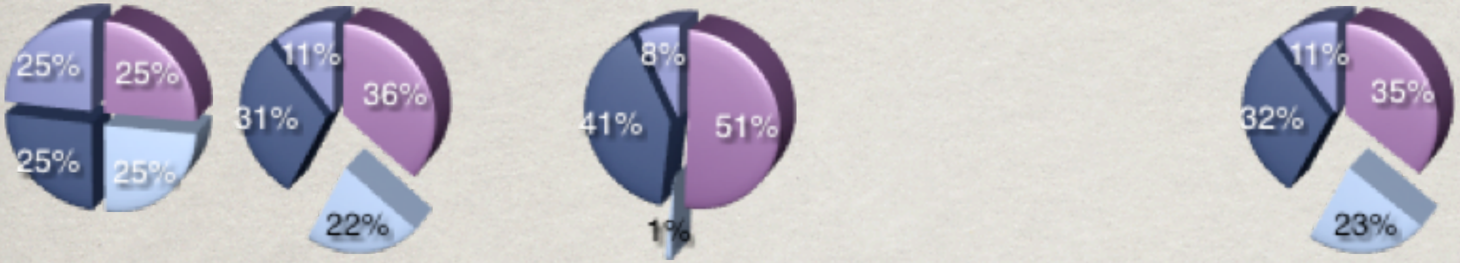


P3@868MHz P4@2.5GHz P4@2GHz P4@2.8GHz

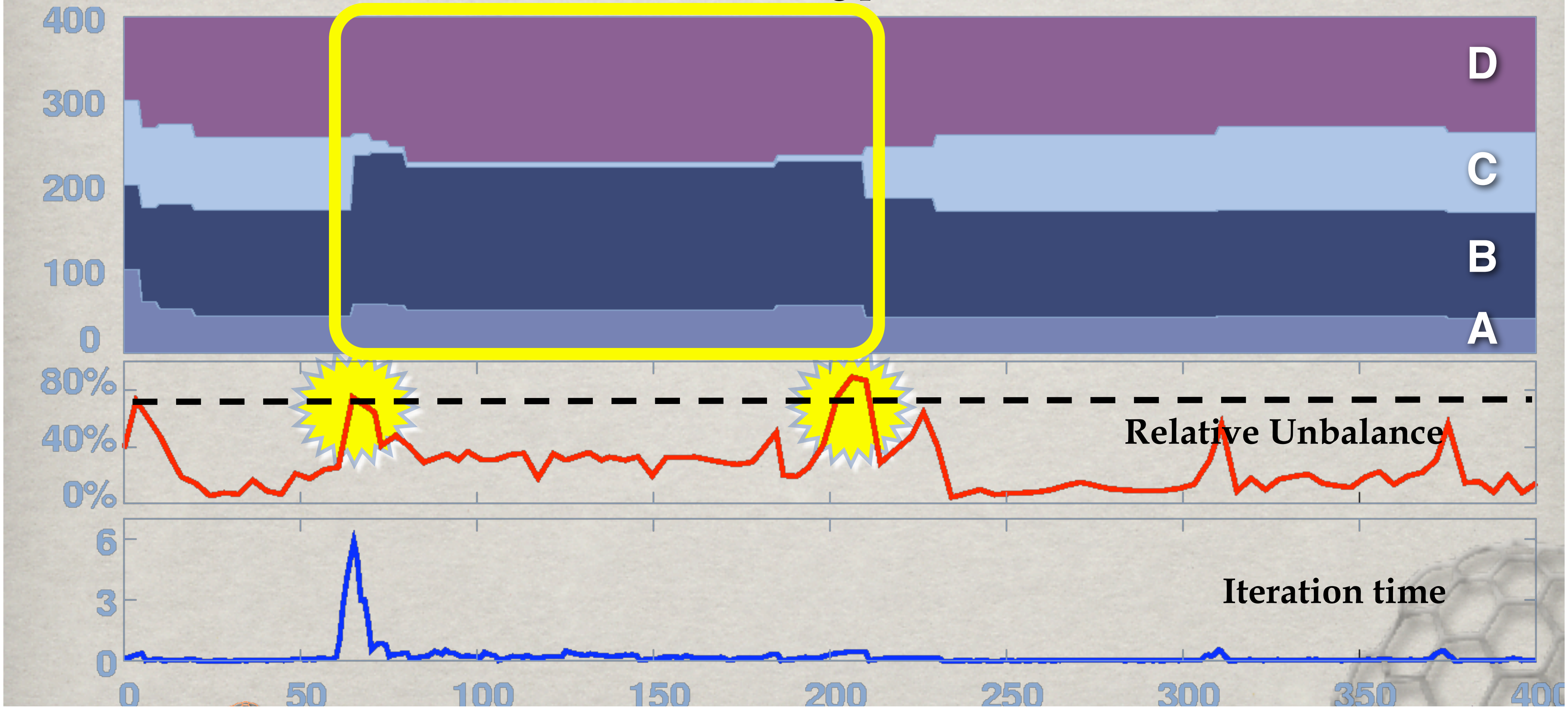
Expected work balance among platforms



Exp 2: Data-Parallel(STP)



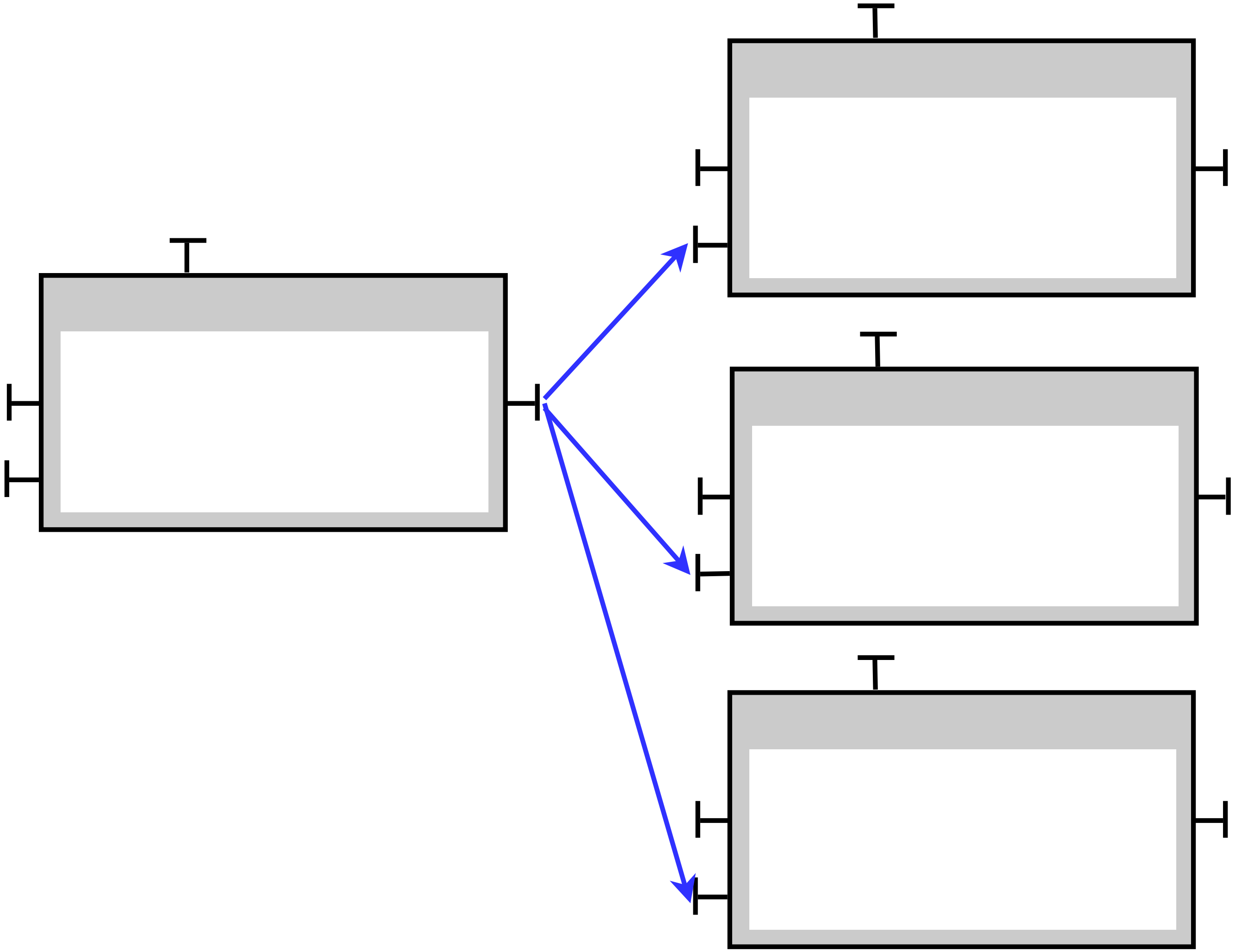
Distribution of load among platforms (n. of VPs)



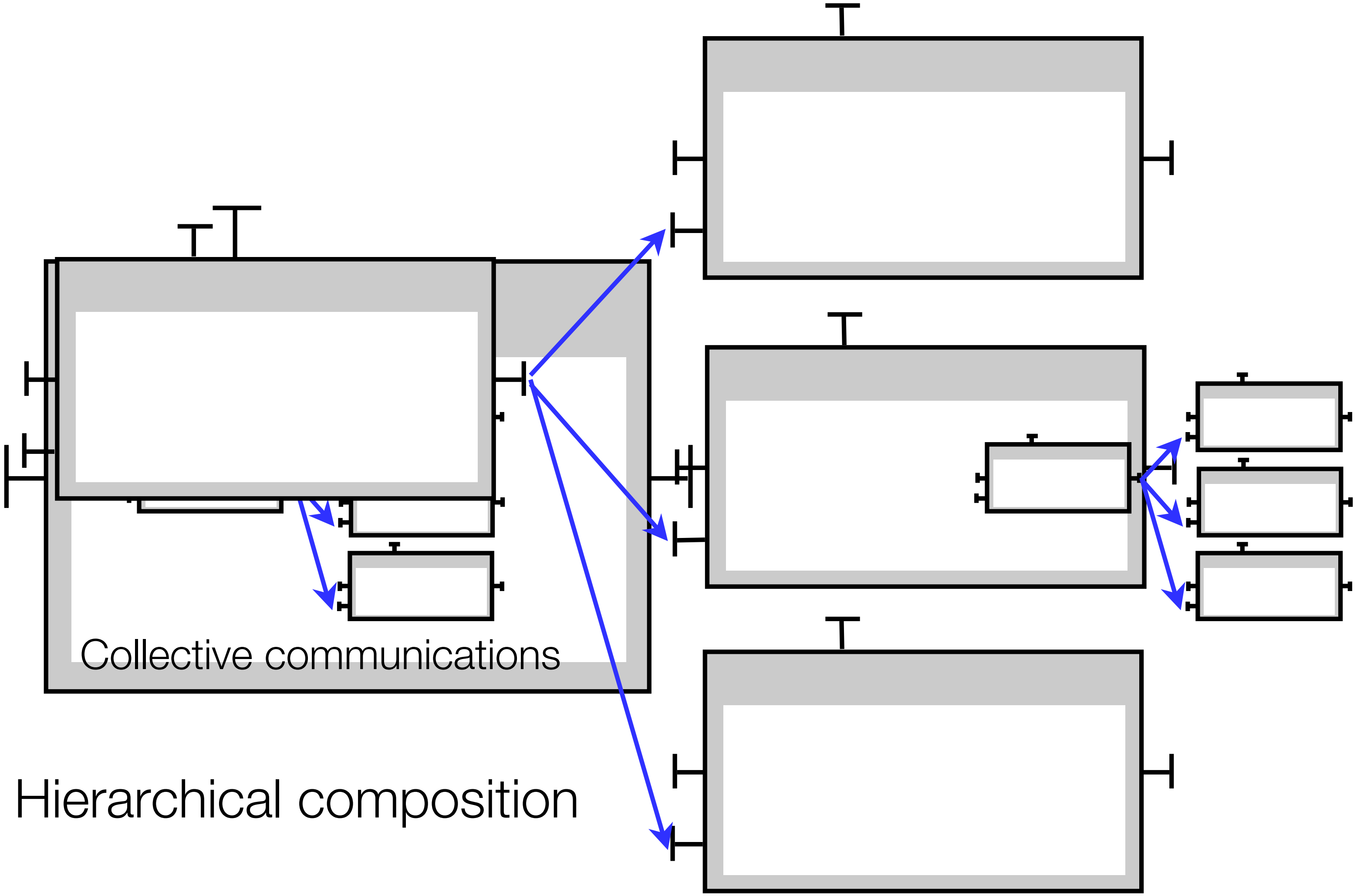
The Programming model Institute perspective

- GCM: the Grid Component Model
  - hierarchical composition
  - RPC + data/stream + event ports + structured interaction patterns
  - component autonomic control (non functional features, including performance, and grid management)
  - reference implementation in GridCOMP
  - interoperable (WS),
    advanced programming models on top of GCM (skeletons)

The GCM features: collective communications

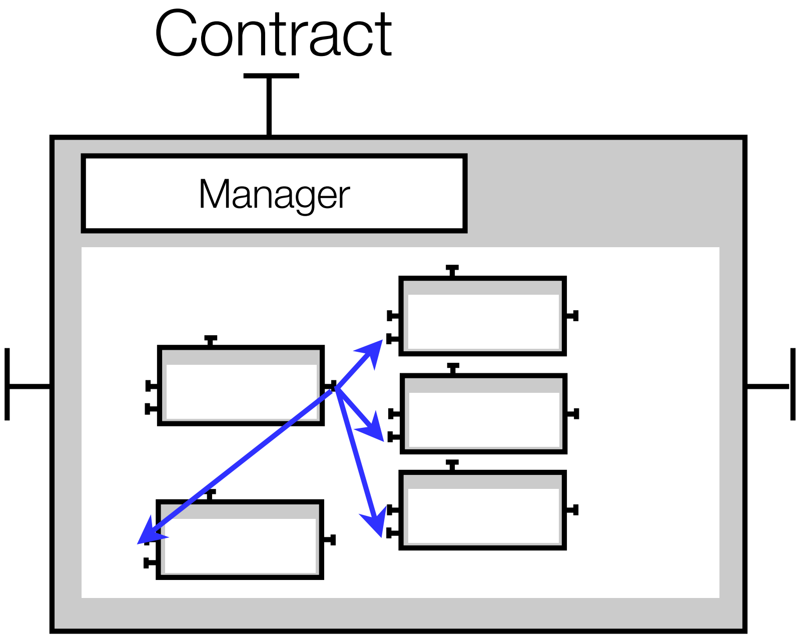


The GCM features: hierarchical composition

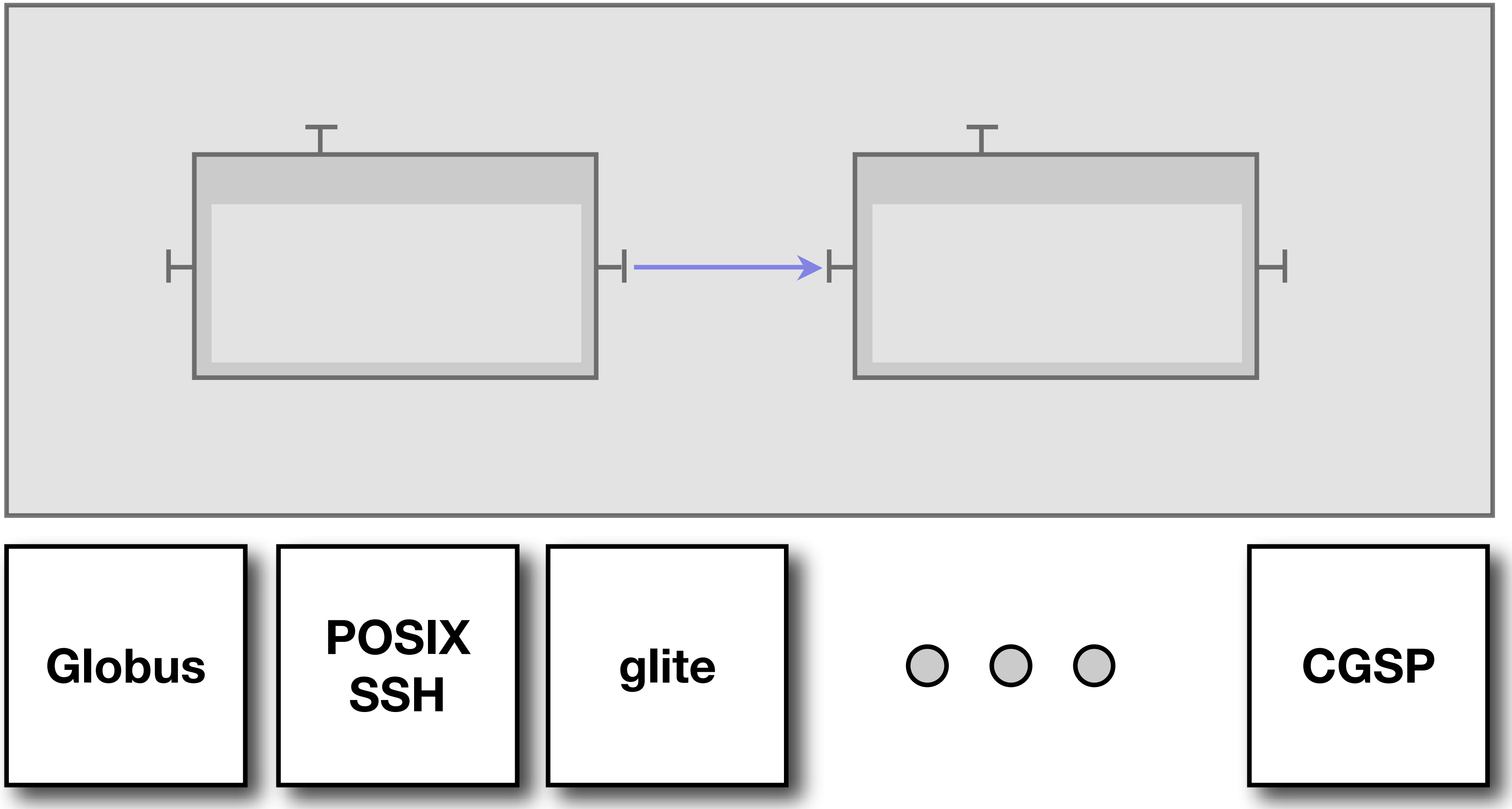


Hierarchical composition

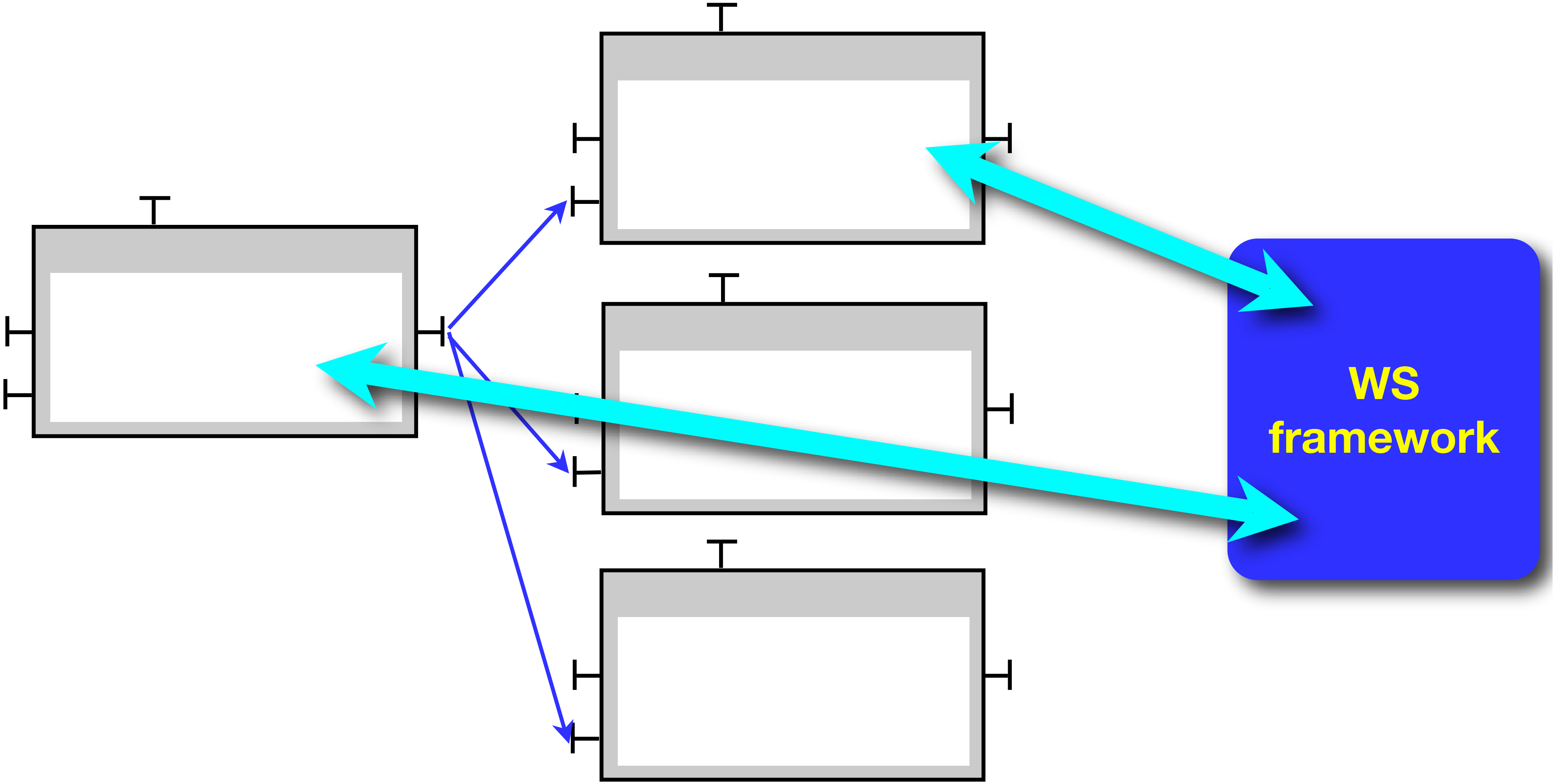
The GCM features: autonomic management



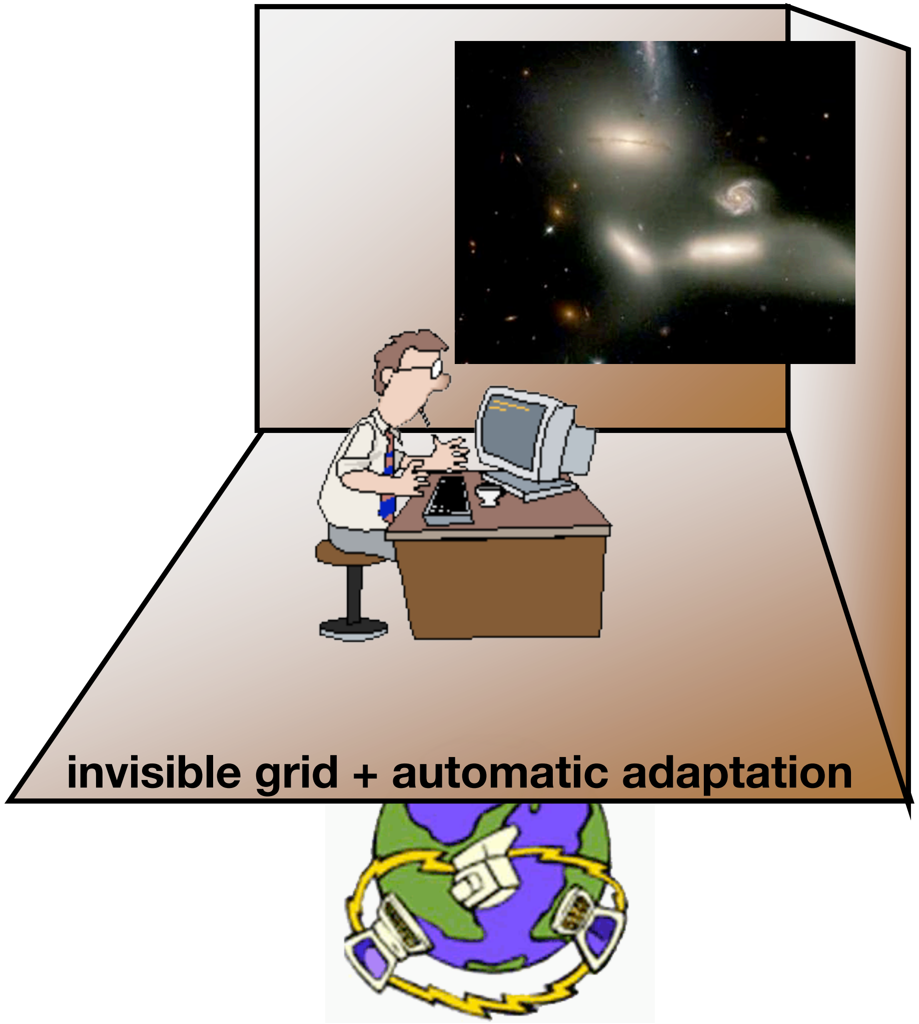
The GCM features: portability



The GCM features: interoperability

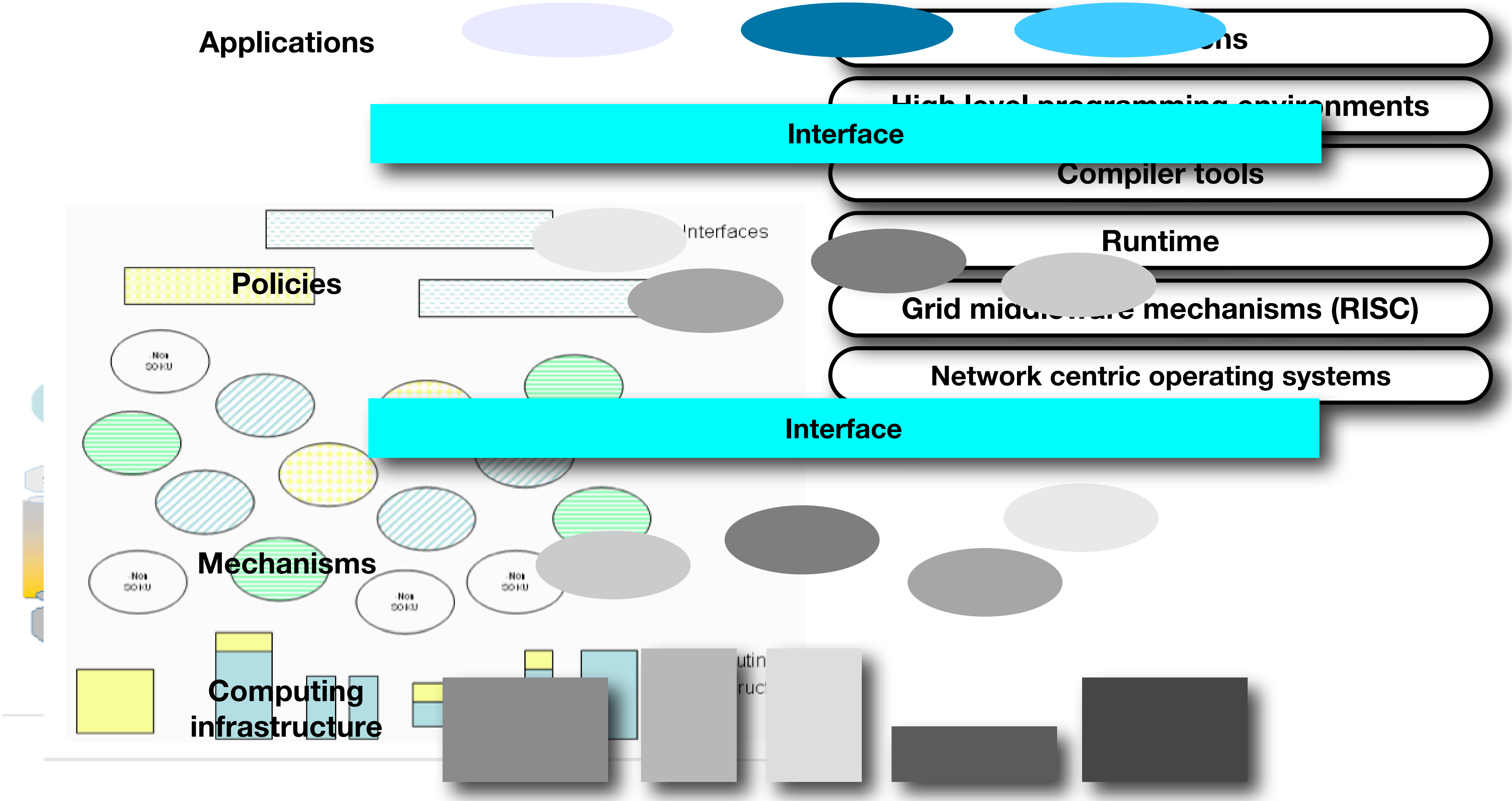


The challenges: “invisible” vs “aware” models



- Actually
  - Vision and awareness of grid peculiarities and problems (system programmer level)
  - Invisible grid á la power grid (application programmer level)

The challenges: bunch of services vs. structured



Applications

Interface

Interfaces

Policies

Mechanisms

Computing infrastructure

Interface

Applications

High level programming environments

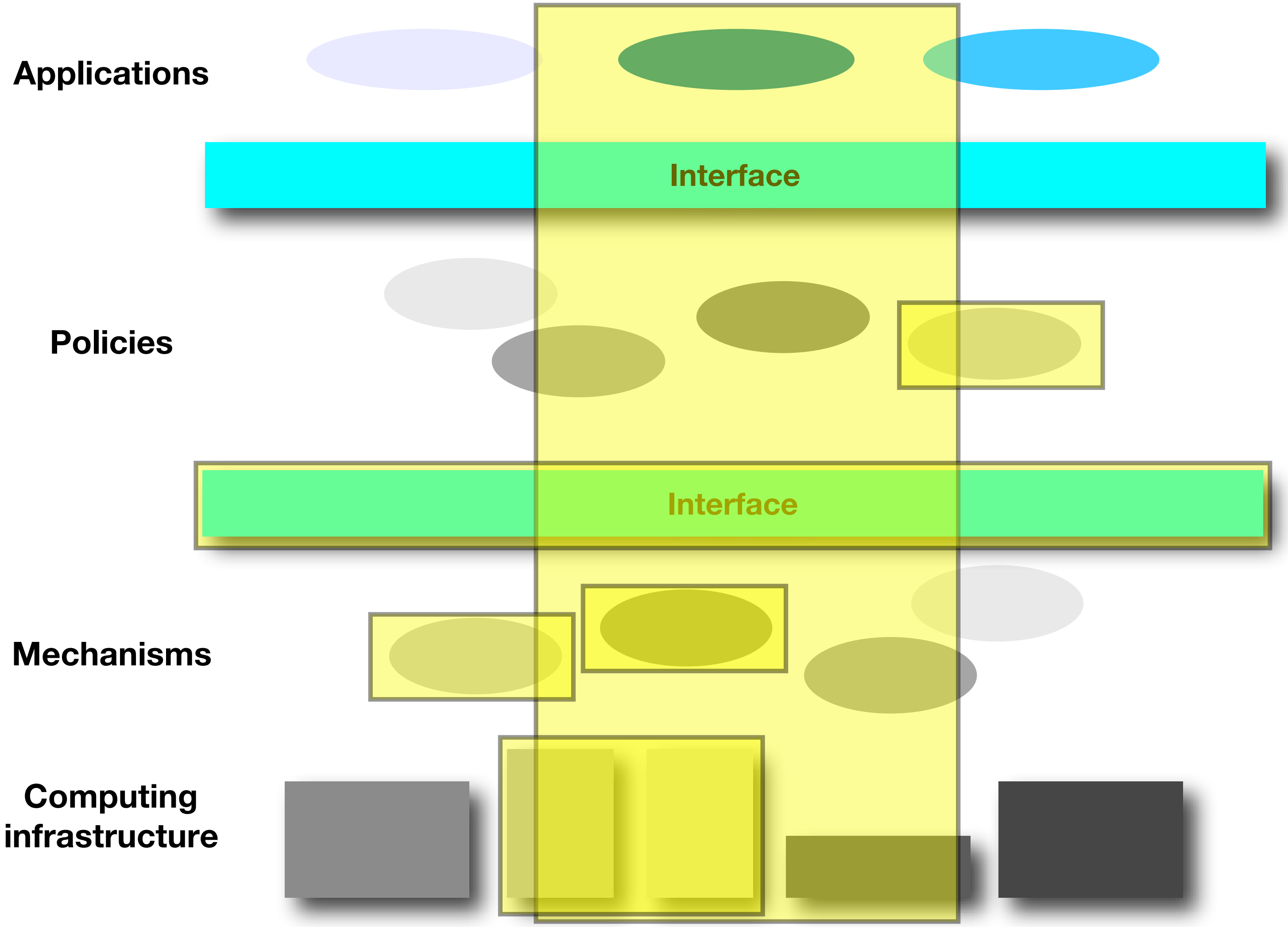
Compiler tools

Runtime

Grid middleware mechanisms (RISC)

Network centric operating systems

The challenges: AOP vs. everywhere concerns



The challenges: global vs. grid computing ???

- is there any difference among global and grid computing ?
  - if we get rid of terminology issues
  - if we choose each time a common abstraction level
  - if take the more general/abstract viewpoint possibile