

# JuxMem: a fault-tolerant grid data-sharing service for scientific applications

Gabriel Antoniu, Luc Bougé, Loïc Cudennec, Mathieu Jan,  
Sébastien Monnet

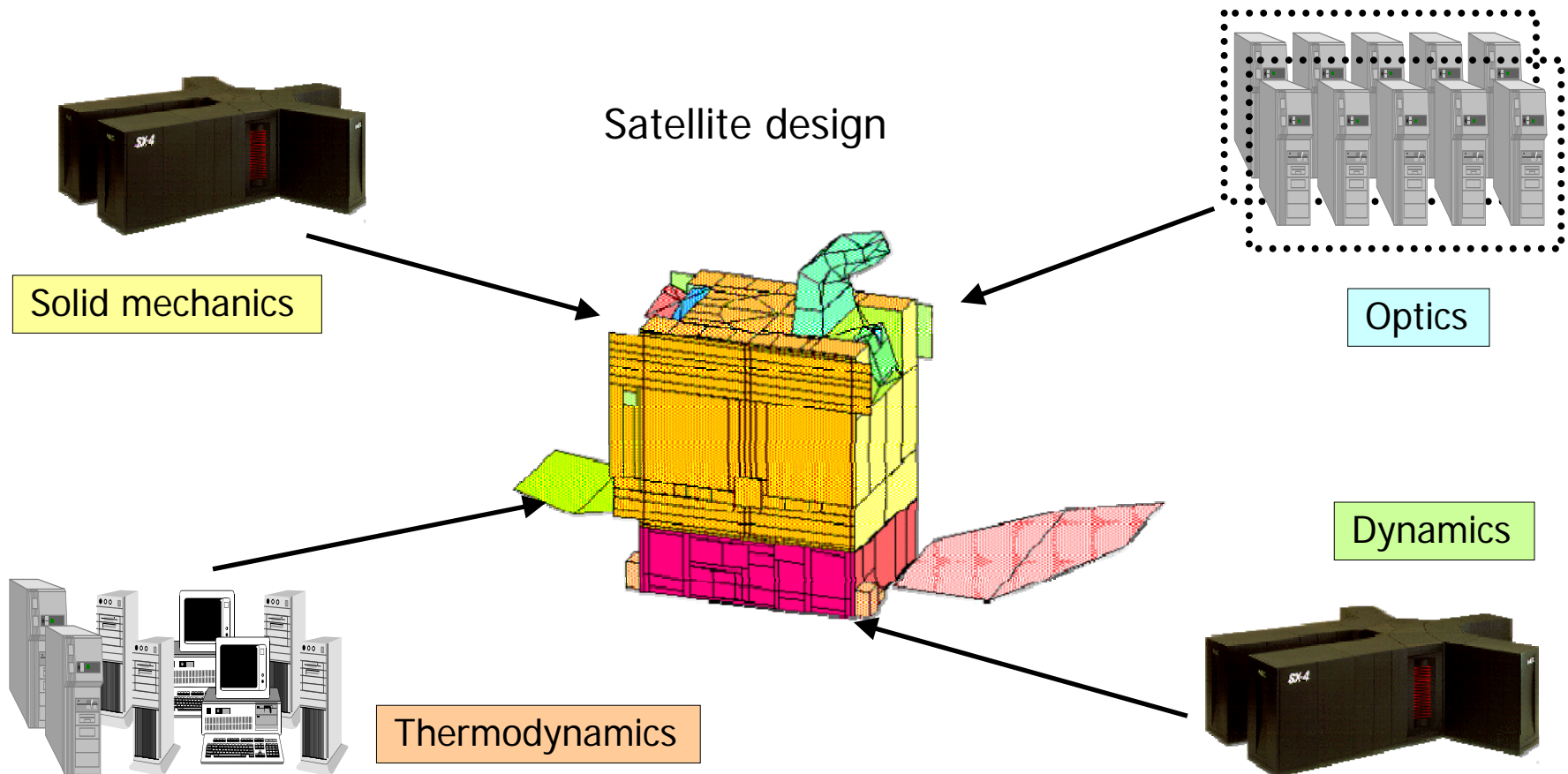
Phenix workshop

Rennes - December 02 2005



# Context: Data Management on the Grid

- Distributed numerical simulations (code coupling)
  - Problem: data management



# Current approaches: explicit data management

- Explicit data localization and transfer
  - GridFTP [ANL], MPICH-G2 [ANL]
    - Security, parallel transfers
  - Internet Backplane Protocol [UTK]



## ■ Limitations

- No **transparency** => Increased complexity at large scale
- No **consistency** guarantees for replicated data



# Handling transparency & consistency: Distributed Shared Memory systems

## ■ Features

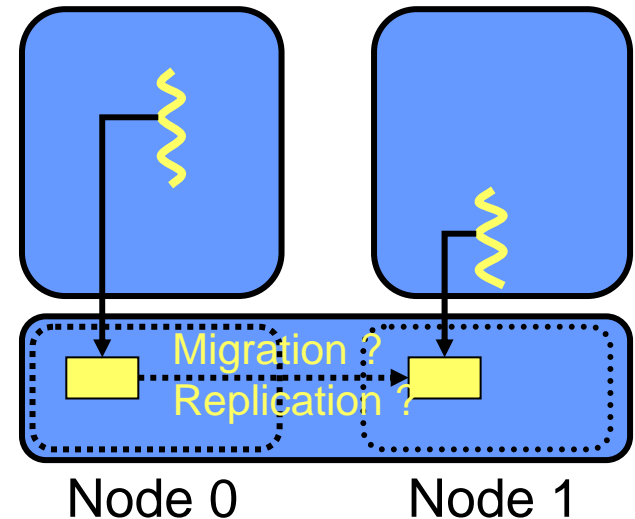
- Uniform access to data via a global identifier
- **Transparent** data localization and transfer
- **Consistency** models and protocols

## ■ But

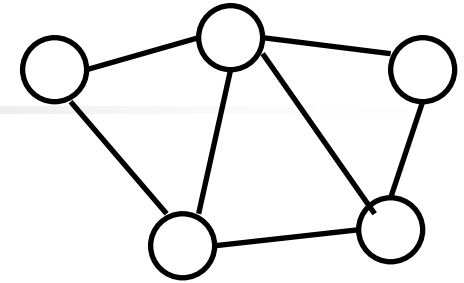
- Small-scale, static architectures

## ■ Challenge on a grid architecture

- Integrate new hypotheses !
  - **Scalability**
  - **Dynamic nature**
  - **Fault tolerance**



# Data sharing at a large scale: peer-to-peer systems



## ■ Features

- Excellent **scalability**: millions of nodes
- High **volatility tolerance**

## ■ But

- Sharing read-only data (mostly)
- Few exceptions: Ivy [MIT], Oceanstore [UCB] ,  
Pastis [LIP6, France]

## ■ Question

- What consistency models and protocols for a grid environment?



# DSM systems and P2P systems

- Comparing basic hypotheses

	<b>DSM</b>	<b>P2P</b>
<b>Scale</b>	$10^1$ - $10^2$	$10^5$ - $10^6$
<b>Dynamicity</b>	Null	High
<b>Resource homogeneity</b>	Homogeneous (clusters)	Heterogeneous (Internet)
<b>Control and trust</b>	High	Low
<b>Topology</b>	Flat	Flat
<b>Data type</b>	Mutable	Immutable
<b>Typical applications</b>	Scientific computation	File sharing and storage

# Idea: Data Sharing Service

- Proposal: hybrid approach
  - **DSM systems**: consistency and transparent access
  - **P2P systems**: scalability and high dynamicity

	<b>DSM</b>	<b>Grid Data Service</b>	<b>P2P</b>
<b>Scale</b>	$10^1$ - $10^2$	$10^3$ - $10^4$	$10^5$ - $10^6$
<b>Dynamicity</b>	Null	Medium	High
<b>Resource homogeneity</b>	Homogeneous (clusters)	Rather heterogeneous (clusters of clusters)	Heterogeneous (Internet)
<b>Control and trust</b>	High	Medium	Low
<b>Topology</b>	Flat	Hierarchical	Flat
<b>Data type</b>	Mutable	Mutable	Immutable
<b>Typical applications</b>	Scientific computation	Scientific computation and data storage	File sharing and storage

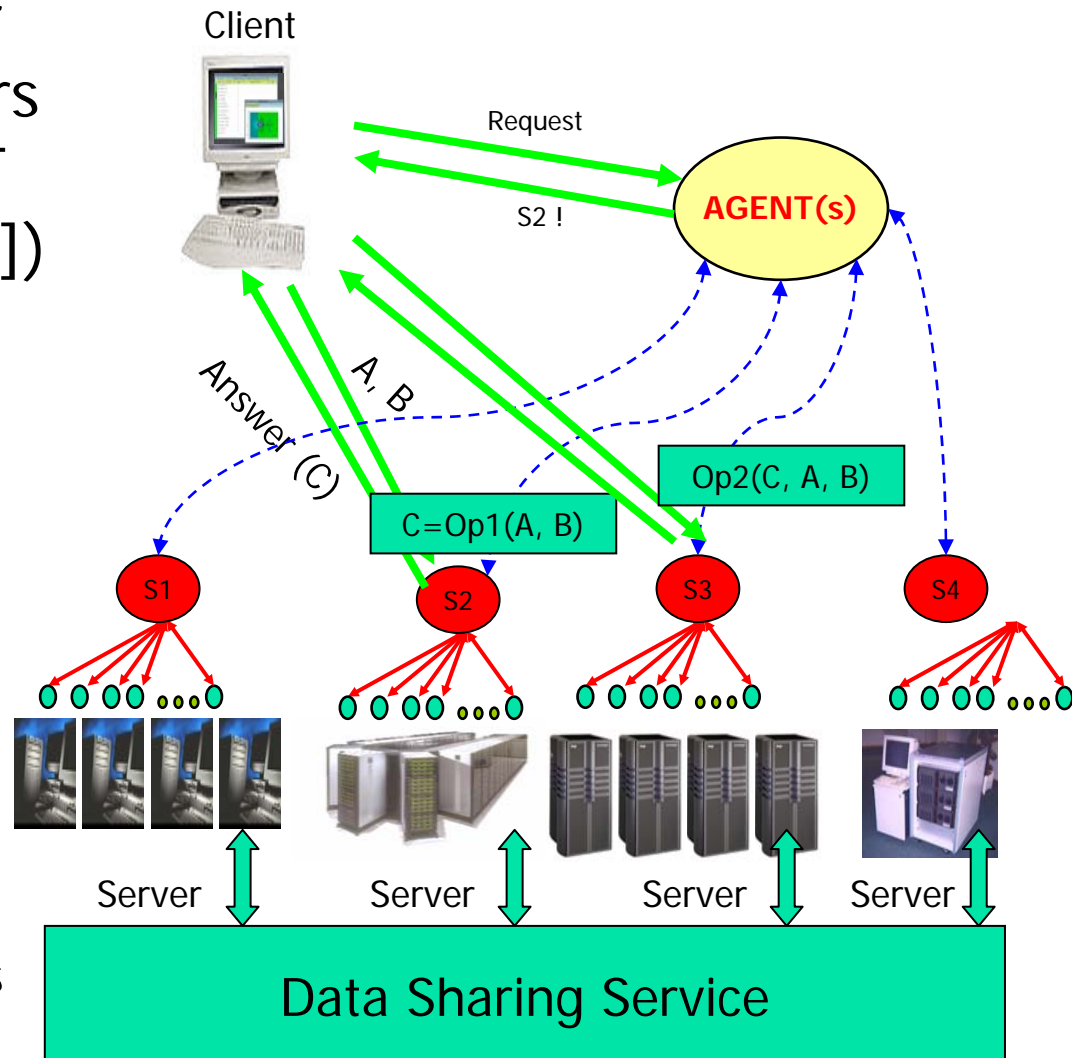
# Why such a service?

- Data sharing service for Network Enabled Servers (e.g. the Grid-RPC DIET environment [ENS Lyon])

- Data persistence
- Transparent localization
- Consistency
- Fault tolerance
- Automatic redistribution

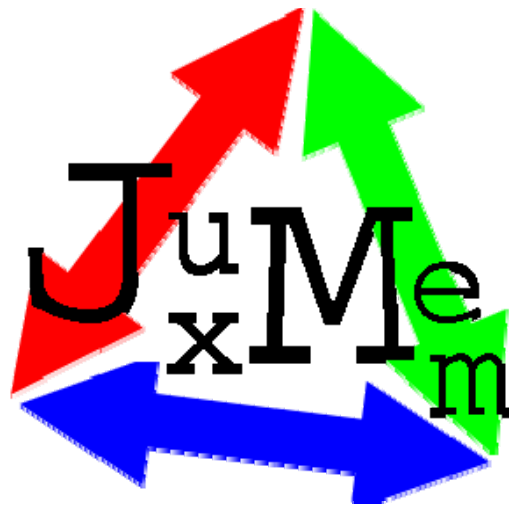
- Motivating application: Grid-TLSE

- Portal for solving matrices-based problems

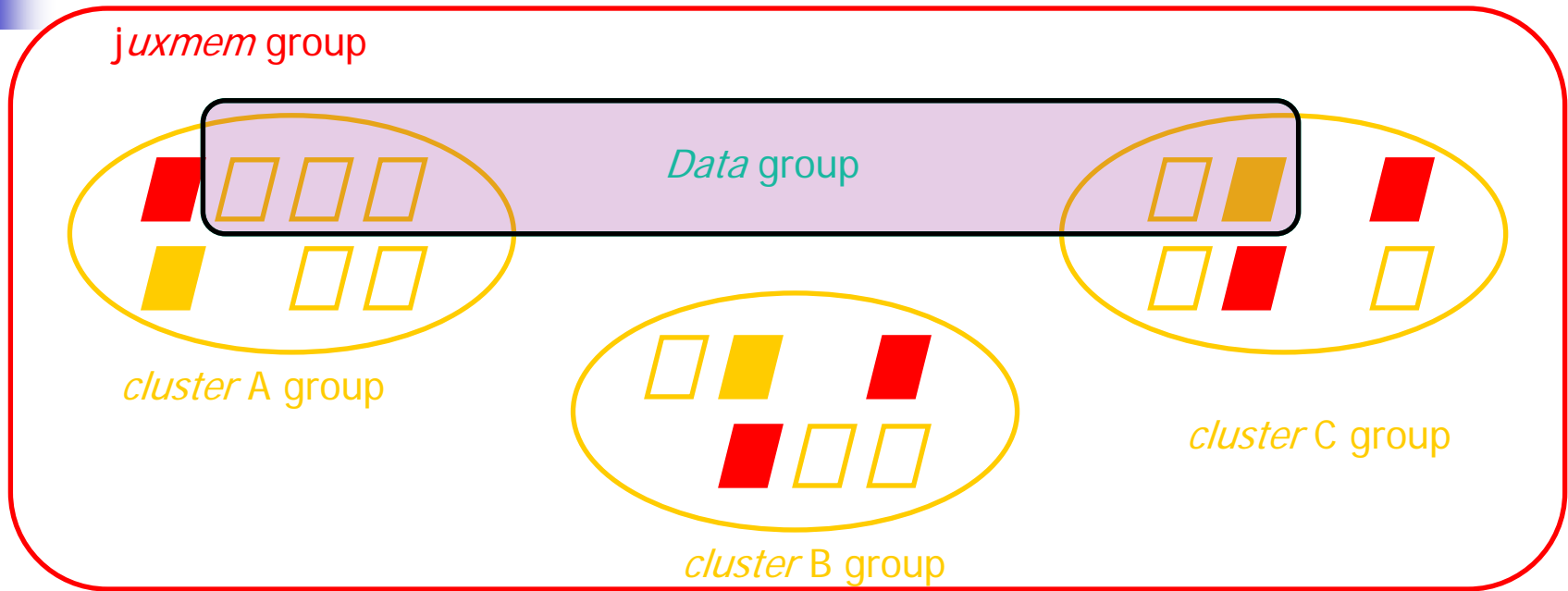


JuxMem:  
an architecture proposal & implementation  
of a grid data-sharing service

---

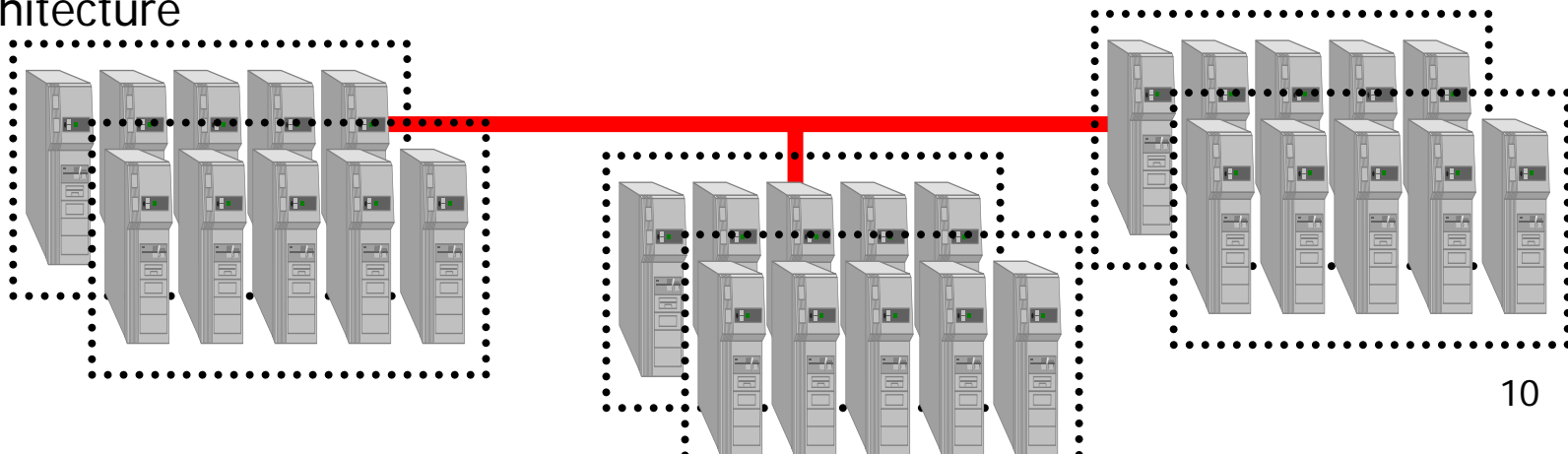


# JuxMem: an Architecture Proposal



Logical architecture

Physical architecture





# Goals

---

- Ensure data availability even in presence of failure
- Ensure data consistency

- JuxMem core
  - ID
  - Communications (send / receive)
  - Publish / search (local and global)



**JuxMem core: juk  
(Mathieu Jan)**





# Assumptions

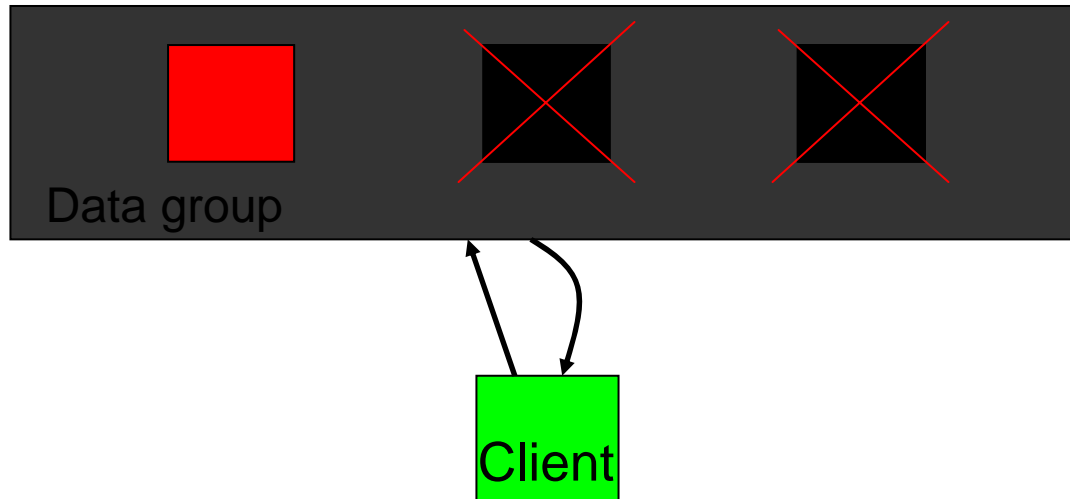
---

- Failures
  - Fail-stop
  - Fairly lossy channels
- Unknown upper bounds for
  - Transfer delay
  - Computational time

Goal:

Data persistency

- Needs: fault tolerance mechanisms
  - Technique: replication



# Replication needs

- Transparency to use
- Self-organization / adaptability
- Performance

**Self-organizing group**

**JuxMem core: juk  
(Mathieu Jan)**

**JXTA™**   
**CONNECTED**

# JuxMem layers

**Self-organizing group  
provider**

**Self-organizing group  
client stub**

**JuxMem core: juk  
(Mathieu Jan)**





# Group membership

---

- Maintain up-to-date (synchronize) the members lists
- API:
  - join(ID id)
  - leave(ID id)
  - send(Message m)

Group membership



# Atomic multicast

---

- Every members deliver the same messages, in the same order
  - View synchrony
  - active (pessimistic) replication
- API : `send(Message m)`

Group membership

Atomic multicast



# Consensus

---

- Agreement problem
  - Messages order
- API:
  - Object decide(Object proposition)
- Need for failure detection

Group membership

Atomic multicast

Consensus



# Failure detectors

---

- Watch all the nodes
- Keep updated a suspect list
- API
  - Callback function : `SuspisionEvent(ID id)`
  - List `getSupectedProcesses()`

Group membership

Atomic multicast

Consensus

Failure detectors

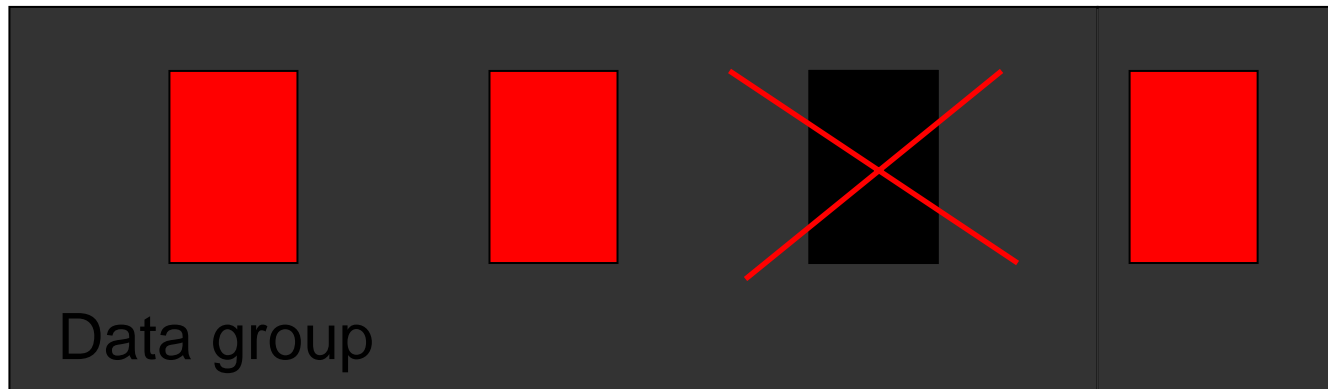


# Hierarchical and adaptable failure detectors

---

- Marin Bertier PhD thesis (LIP6, Paris)
- Scalable
  - Hierarchical
    - All-to-all inside clusters
    - Leader-to-leader among clusters
  - Factorisable
    - Only one message flow shared via an adaptation layer
- Support for varying charge / QoS
  - Adaptability

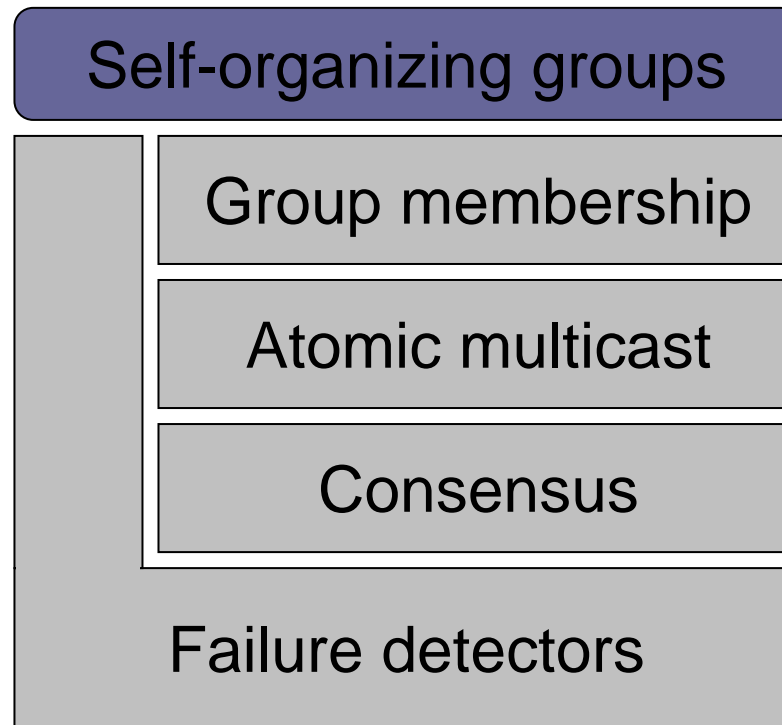
# Self-organizing groups





# Self-organizing groups

---





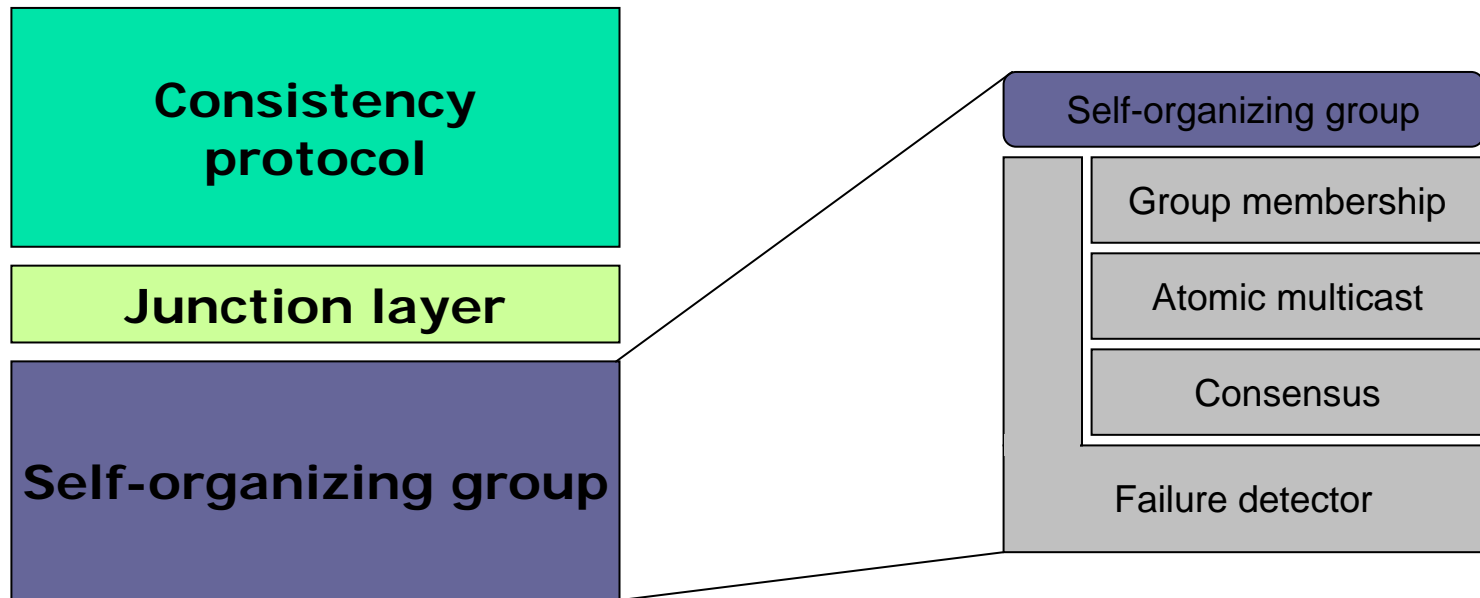
# Data consistency

---

- In grids
  - Explicit data-consistency management
- In DSMs
  - Numerous models and protocols
  - Usually based on stable entities
    - Page manager
    - Home node
    - ...

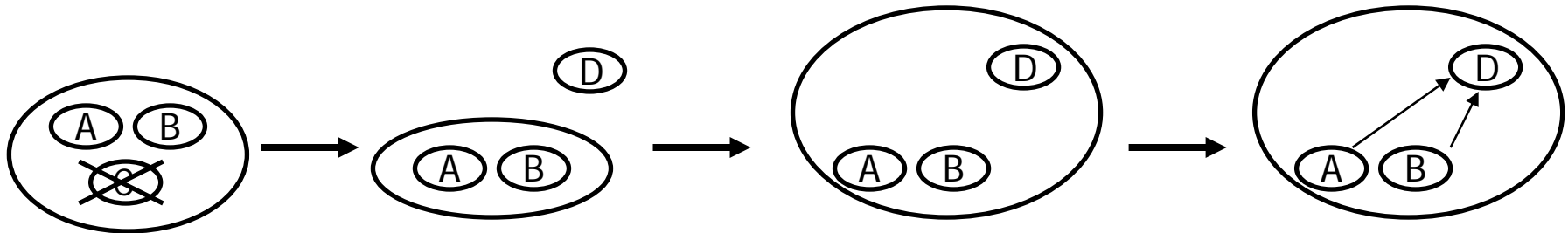
# Data-consistency in a volatile environment

- Principle
  - Stable entity => replication group



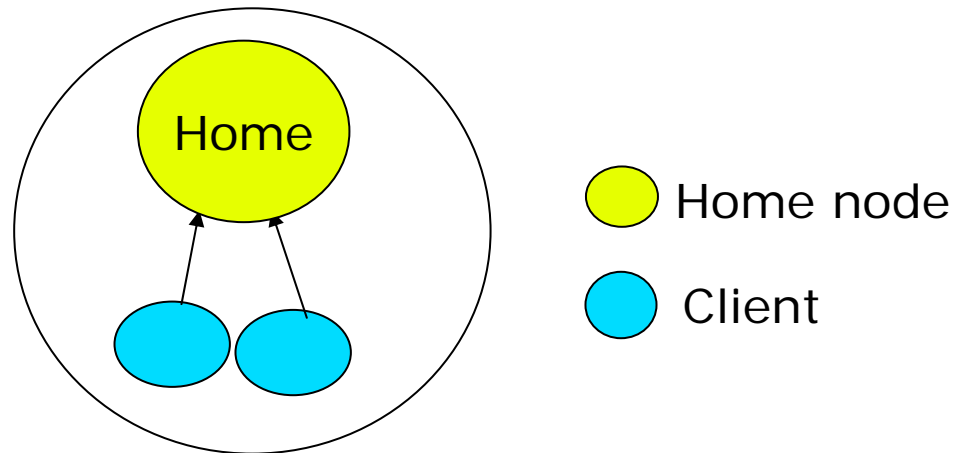
# Interaction

## consistency protocol / replication

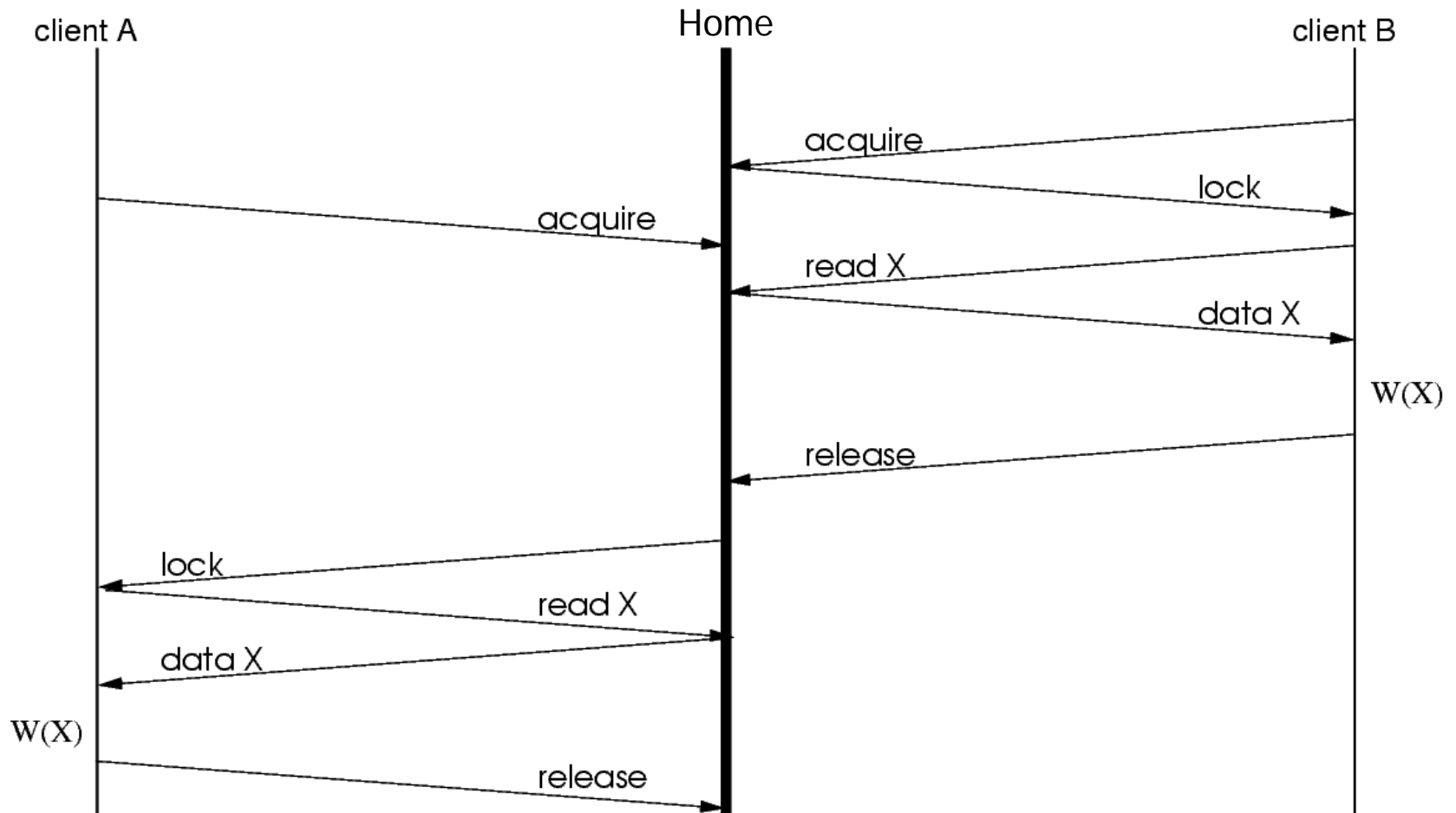


# Building a Fault-Tolerant Consistency Protocol

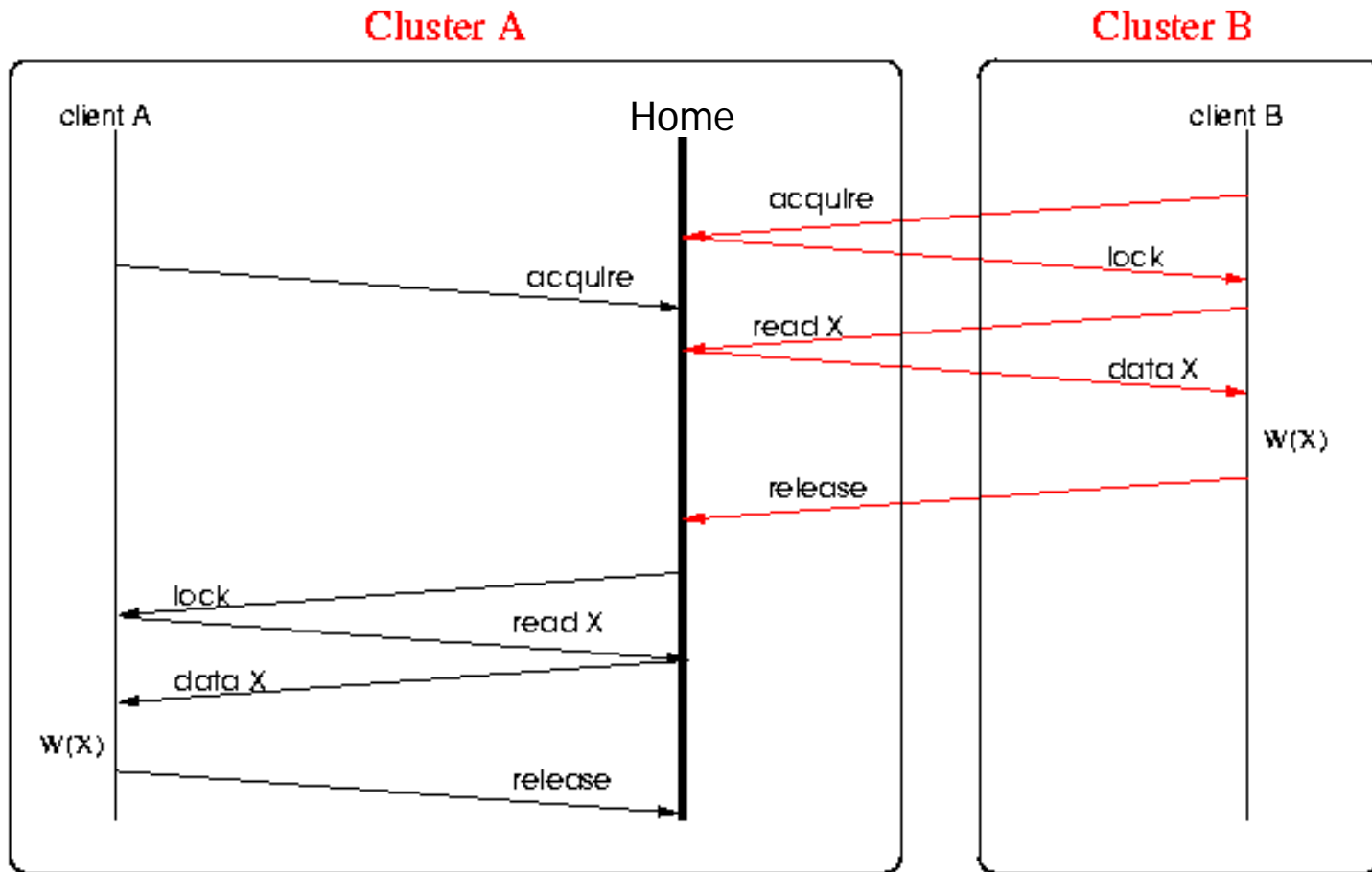
- Starting point: a home-based protocol for entry consistency
  - Relaxed consistency model
    - Explicit association of data to locks
    - MRSW : Multiple Reader Single Writer
      - acquire(L)
      - acquireRead(L)
  - Home-based protocol



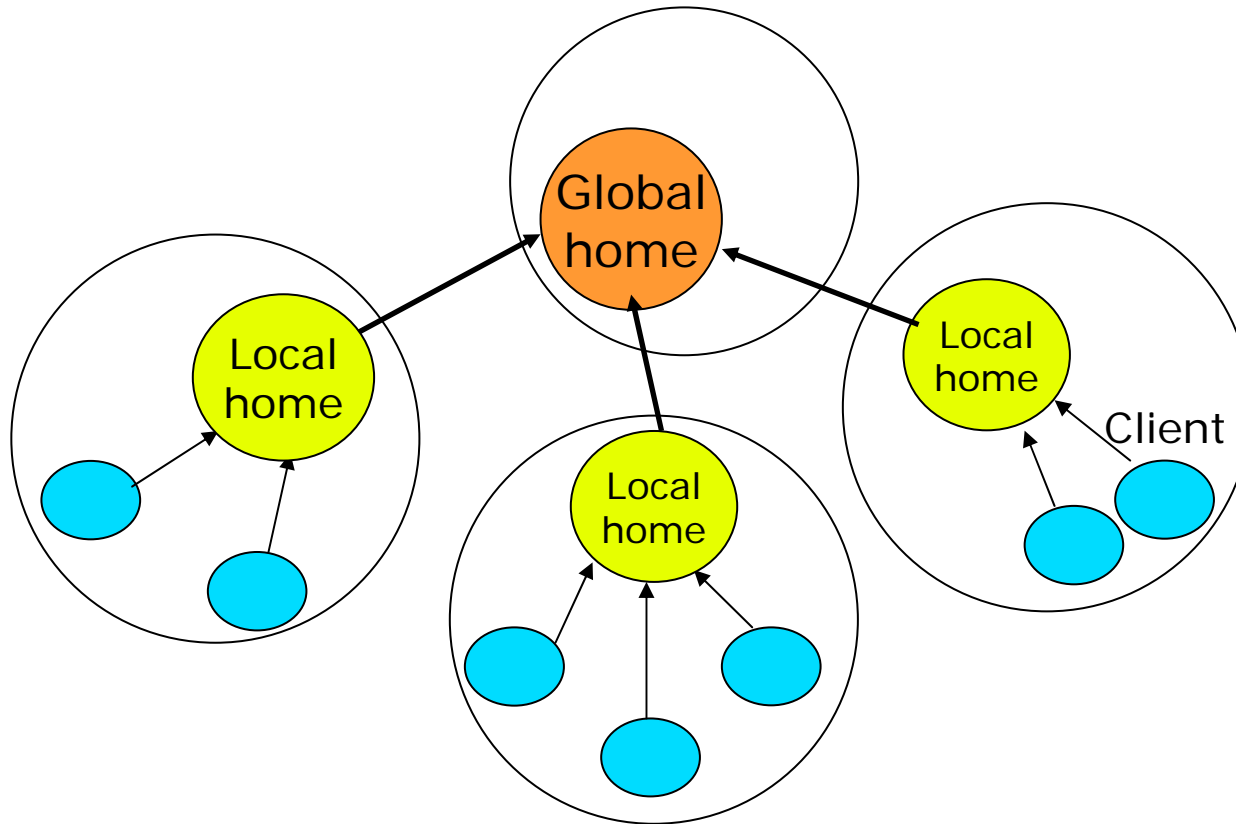
# A Home-Based Protocol Scheme



# Problem: Inter-cluster Latency Higher than Intra-cluster Latency

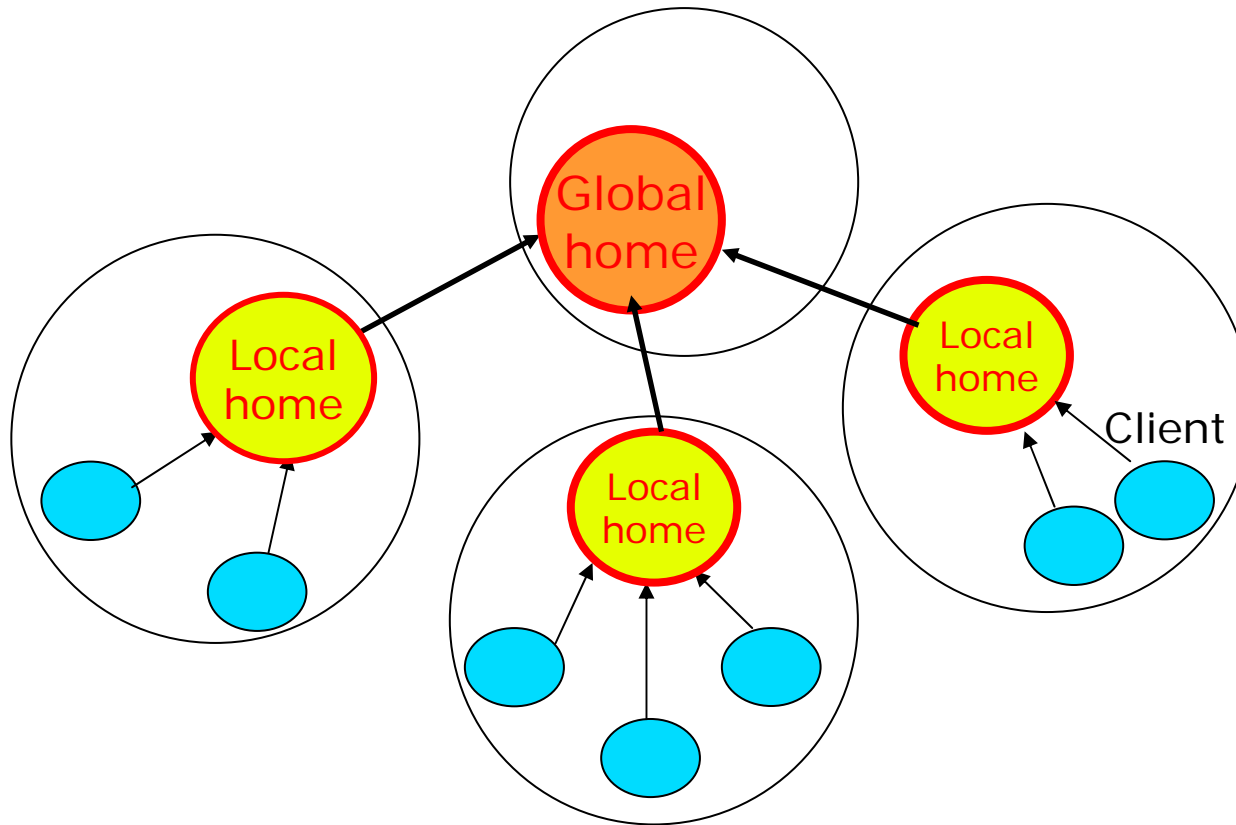


# Next Step: a Hierarchical Consistency Protocol



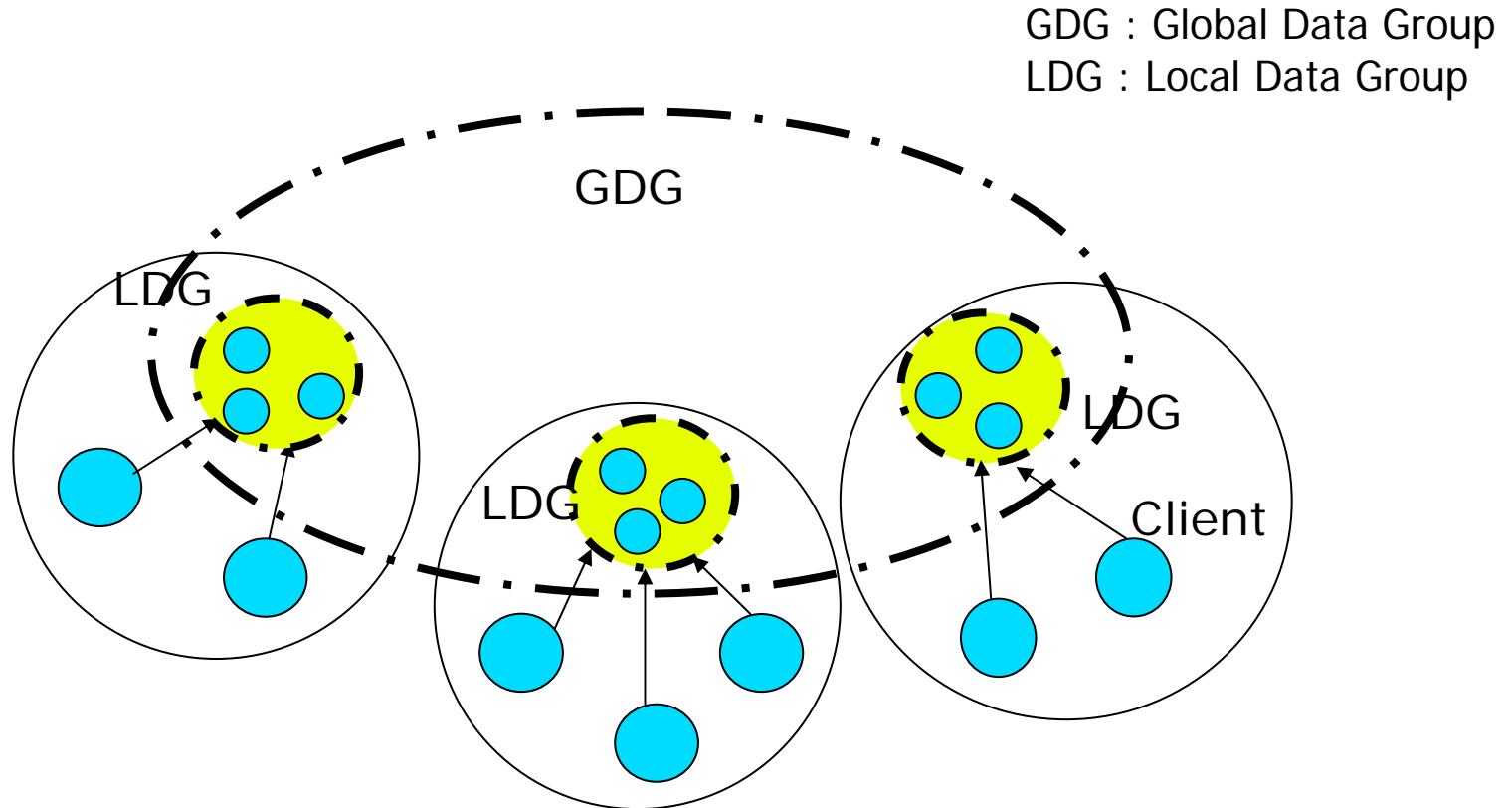
- Inspired by CLRC[LIP6] and H2BRC[PARIS]

# Problem: Critical Entities May Crash



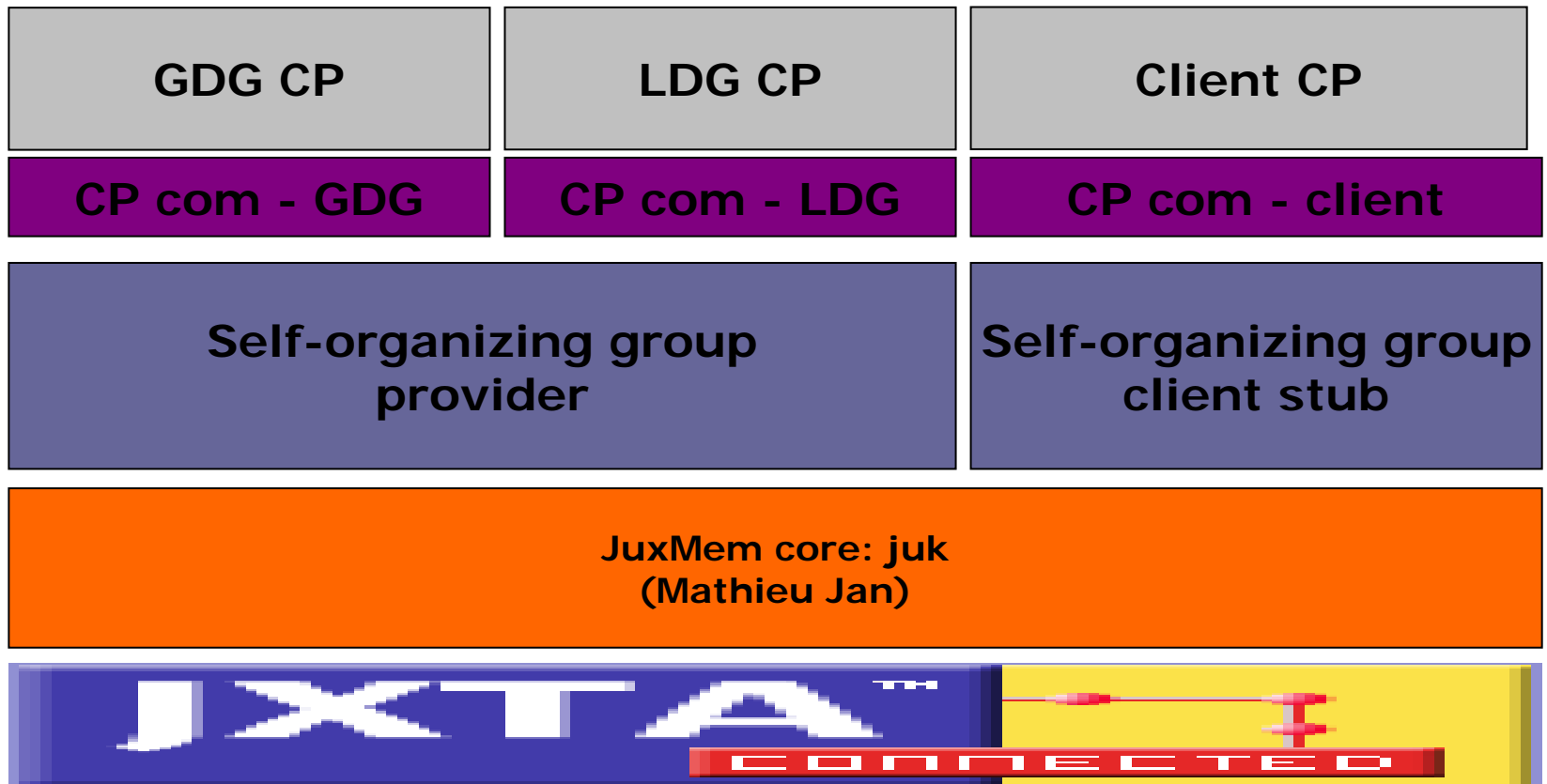
Question: how to support *home* crashes on a grid infrastructure ?

# Solution: Replicate Critical Entities Using Fault-Tolerant Components



- Rely on replication techniques and group communication protocols used in fault-tolerant distributed systems

# JuxMem layers





# JuxMem : a generic architecture

---

- Possibility to choose dynamically
  - Consistency protocol
  - Replication mechanisms
  - Failure detectors
- Currently
  - 3 consistency protocols
  - Pessimistic and optimistic replication



# Conclusion

---

- JuxMem: a hierarchical architecture for a data sharing service for the grid
  - **Hybrid** approach: DSM and P2P systems
  - **Transparent** access to data blocks
  - **Mutable** data
  - **Persistent** storage
  - Active support for peer **volatility**
- Experimental platform for studying the interaction fault-tolerance <-> consistency protocols



# Future work

---

- Implementation and experimentations
  - Multiples failure scenarios
  - Multiples consistency protocols
  - Interactions consistency protocols / fault tolerance layers
- Introspection and adaptation (consistency, fault tolerance) wrt
  - Risk level
  - Applications needs
- Collaborations
  - GDS (Grid Data Service) project of ACI MD
  - UIUC (Indranil Gupta)

<http://juxmem.gforge.inria.fr>