Fractal ProActive and a natural extension for the Grid :

Gathercast and Multicast interfaces

Matthieu Morel

(with Françoise Baude, Denis Caromel and Ludovic Henrio)

France Telecom
R&D Division

INRIA - Oasis team

Fractal workshop

November 29th 2005

Context : Grid computing

- Federation of many computational and storage resources
  - Multiple administrative domains
  - Remote communications
  - Heterogeneous resources and software
- Goal: solving large scale computation problems
  - Science : High Energy Physics (CERN), biology, astrophysics…
  - Business : emerging
- Multiparty communications, parallel computing
  - ⇨ Collective and parallel programming facilities needed
- CoreGrid initiative ⇨ Grid Component Model (GCM)

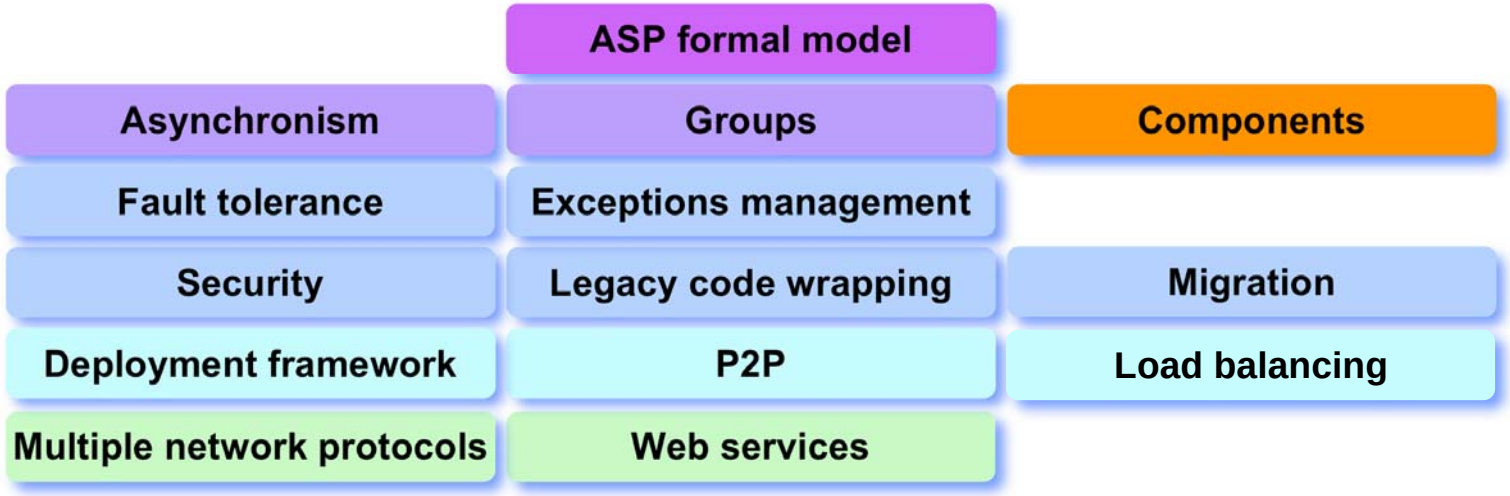
Context : ProActive

- An ObjectWeb project
- A middleware for Grid computing
  - Programming model based on active objects (and components)
  - Asynchronous communications, deterministic model
  - MOP, written in Java
  - Deployment framework
  - Large-scale experiments

    (Grid Plugtests : 2000+ interconnected machines)
- A research project
  - Security, P2P, fault tolerance, exception management, verification, model checking, component-based programming…

Context : ProActive/Fractal

- Implementation of Fractal with ProActive
  - Components are distributed active objects
  - Components can reuse underlying features



  - Standard controllers + customization + interception
  - No sharing
- Common tools : ADL, GUI (… packaging?)

Problem statement

- Multiparty communications
  - 1 → n
  - n → 1
  - n → m
- Communication management
  - Parallelization, synchronization
  - Data redistribution : parameters, results
- Specification of a collective behaviour

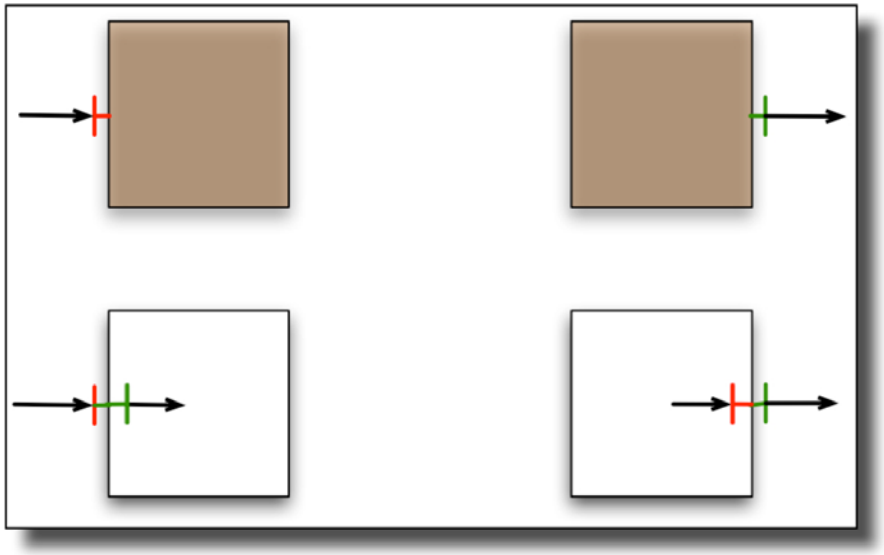
Current situation (Fractal model)

- Simple type system
- Component type ← types of its interfaces
- Interface type :
  - Name
  - Signature
  - Role
  - Contingency
  - Cardinality
- Fractal model is extensible

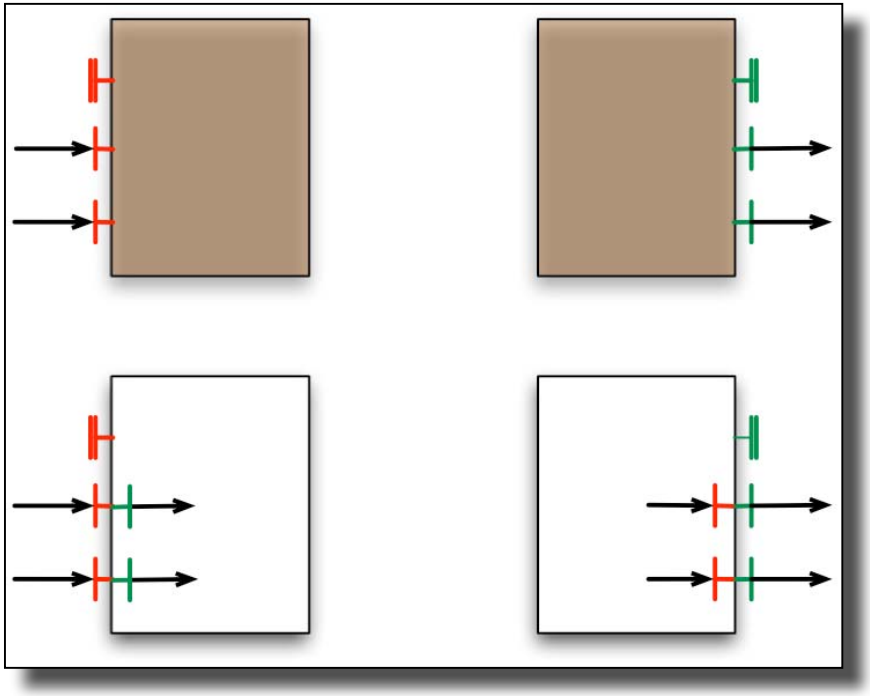
  a Fractal component may provide or use new or alternative control interfaces, type systems, or even component semantics

Current situation : 2 cardinalities

Single



Collection



Current situation: cardinalities

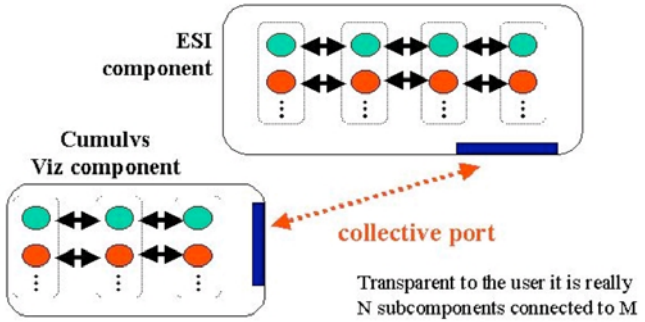
- Single interfaces
- Collection (of) interfaces

} 1 to 1 bindings

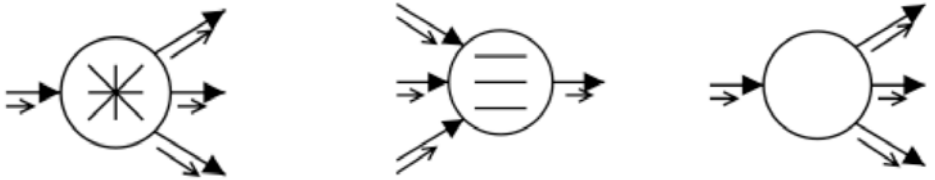
- Binding components
  - ☺ Handle different communication paradigms
    ~ brokers
  - ☹ Complexify the design
  - ☹ Collective behaviour not exposed

Some ideas from the others

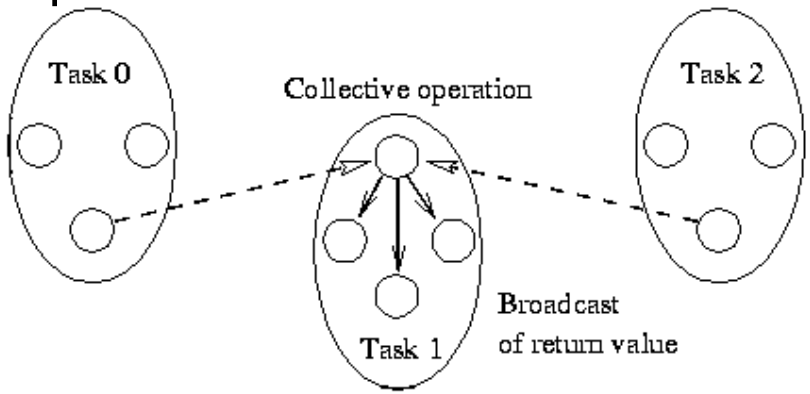
- CCA
  - Collective ports
  - Composable intermediate components
  - PRMI (parallelism + asynchronism)



- ICENI
  - “Tees” : switch, combiner, splitter, gather, broadcast
  - XML data



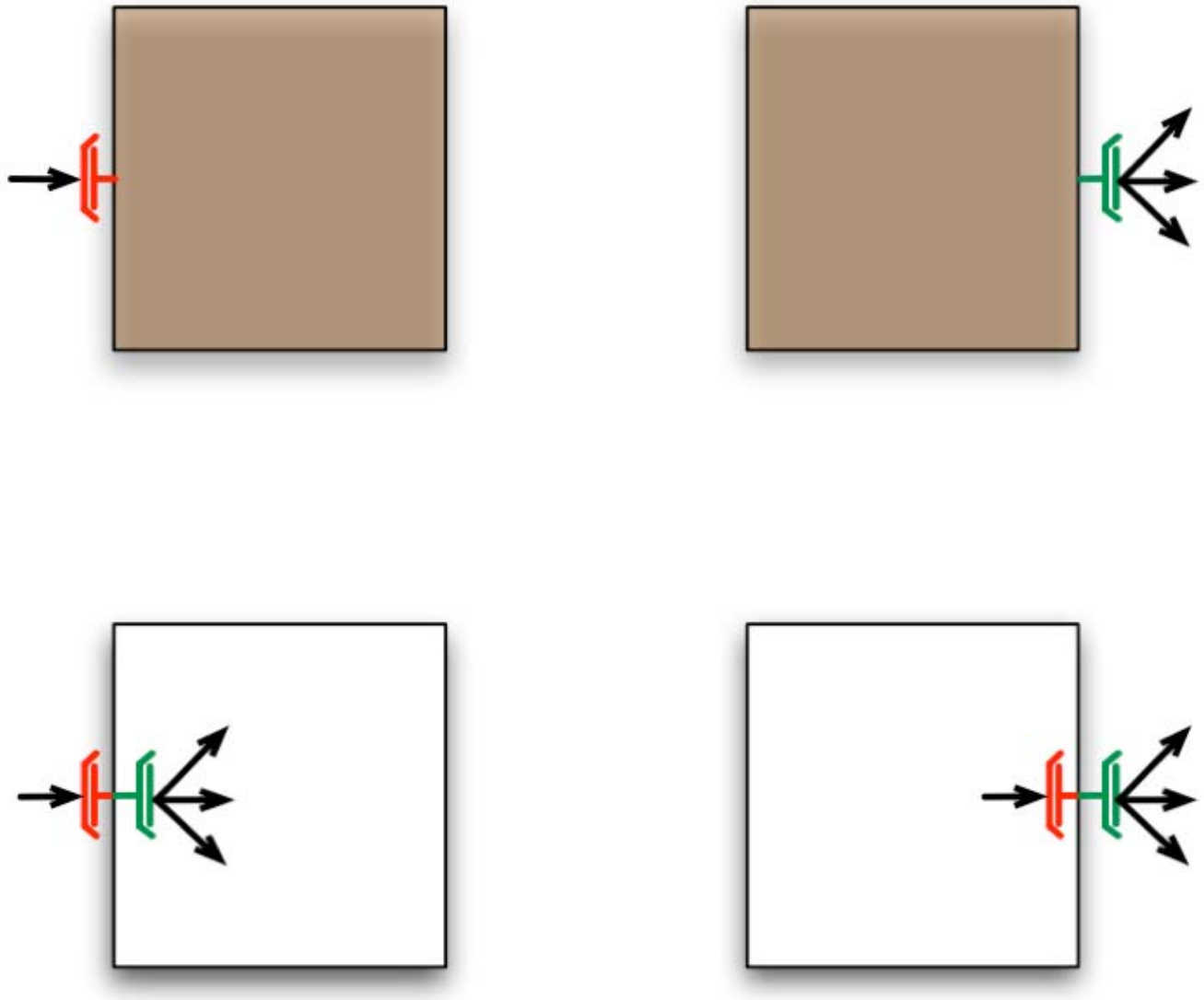
- MPI
  - Communicator groups
  - Multicast
  - Gather
  - Reduce



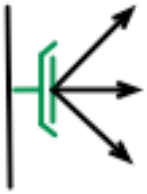
Our proposal

- Simplify the design and configuration of component systems
- Expose the collective nature of interfaces
  - Cardinality attribute
  - Multicast, gathercast, gather-multicast
- The framework handles collective behaviour at the level of the interface
- Based on Fractal API :
  - dedicated controller
  - interface typing

Multicast interfaces

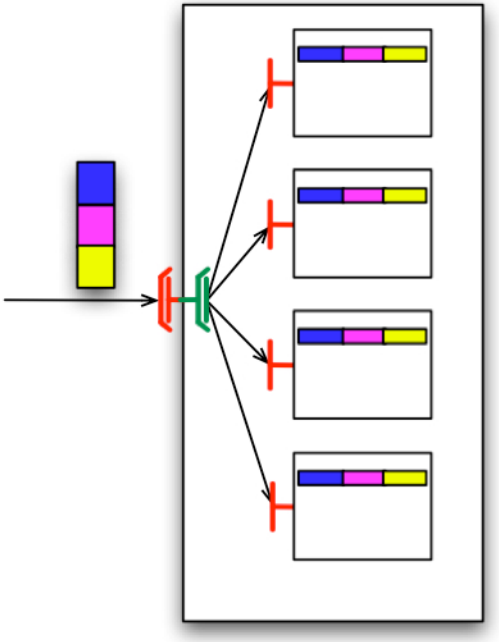


Multicast interfaces

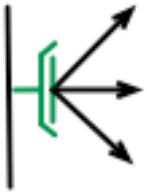


Transform a single invocation into a list of invocations

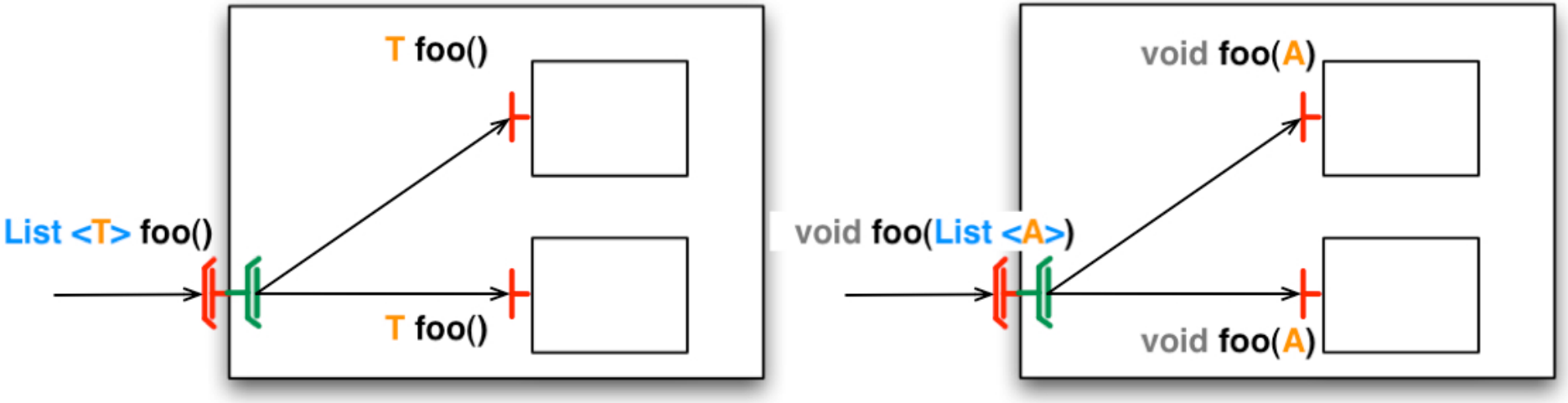
- Multiple invocations
  - Parallelism
  - Asynchronism
  - Dispatch
- Data redistribution (invocation parameters)
  - Parameterisable distribution function
  - Broadcast, scattering
  - Dynamic redistribution (dynamic dispatch)
- Result = list of results



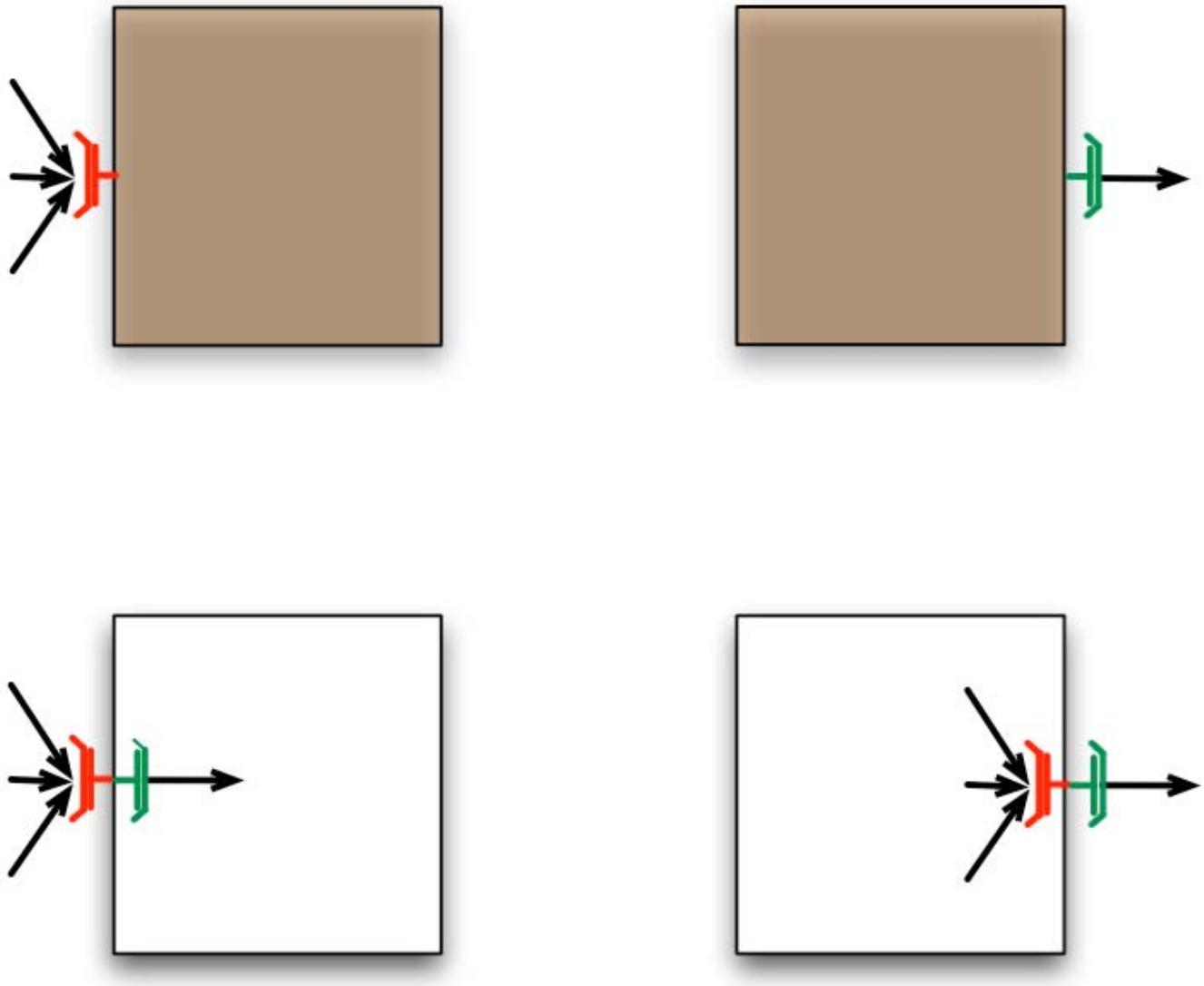
Multicast interfaces



- Results as lists of results
- Invocation parameters may also be distributed from lists



Gathercast interfaces

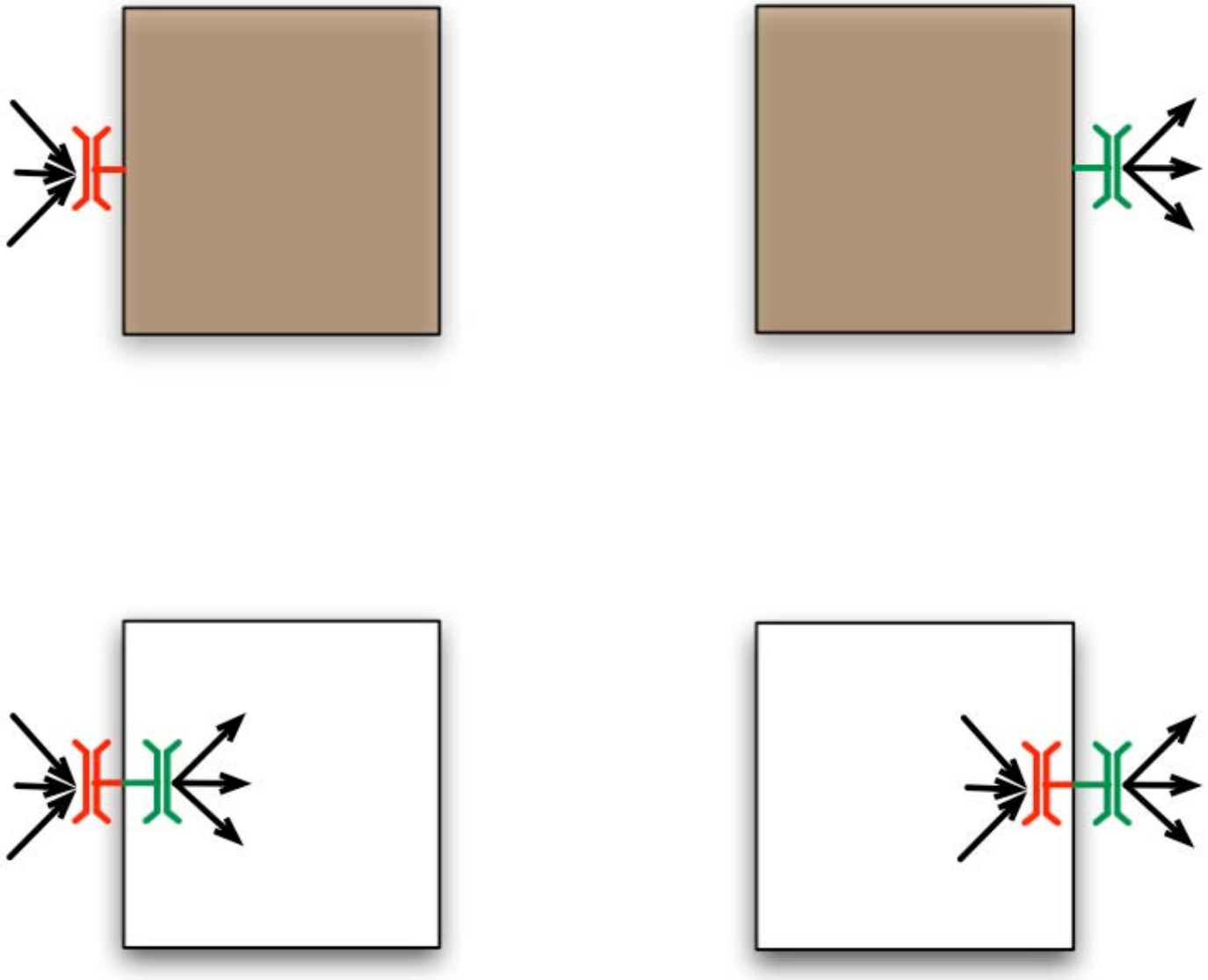


Gathercast interfaces

Transform a list of invocations into a single invocation

- Synchronization of incoming invocations
  - ~ “join” invocations
  - Timeout / drop policy
  - Bidirectional bindings (callers ⇔ callee)
- Data gathering
  - Aggregation of parameters into lists
- Result : redistribution of results
  - Redistribution function

Gather-multicast interfaces



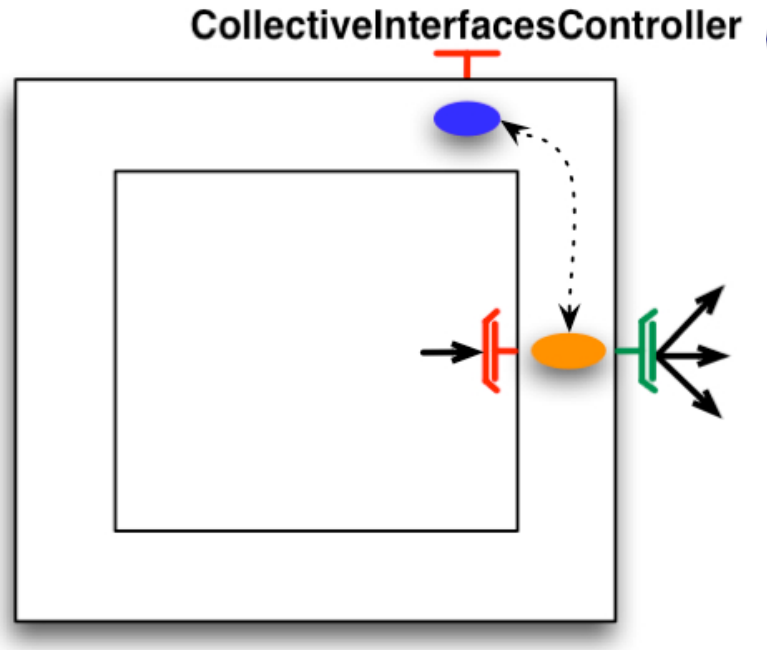
API

- `InterfaceType` interface:

```
string getFcItfCardinality();
```

- `TypeFactory` interface:

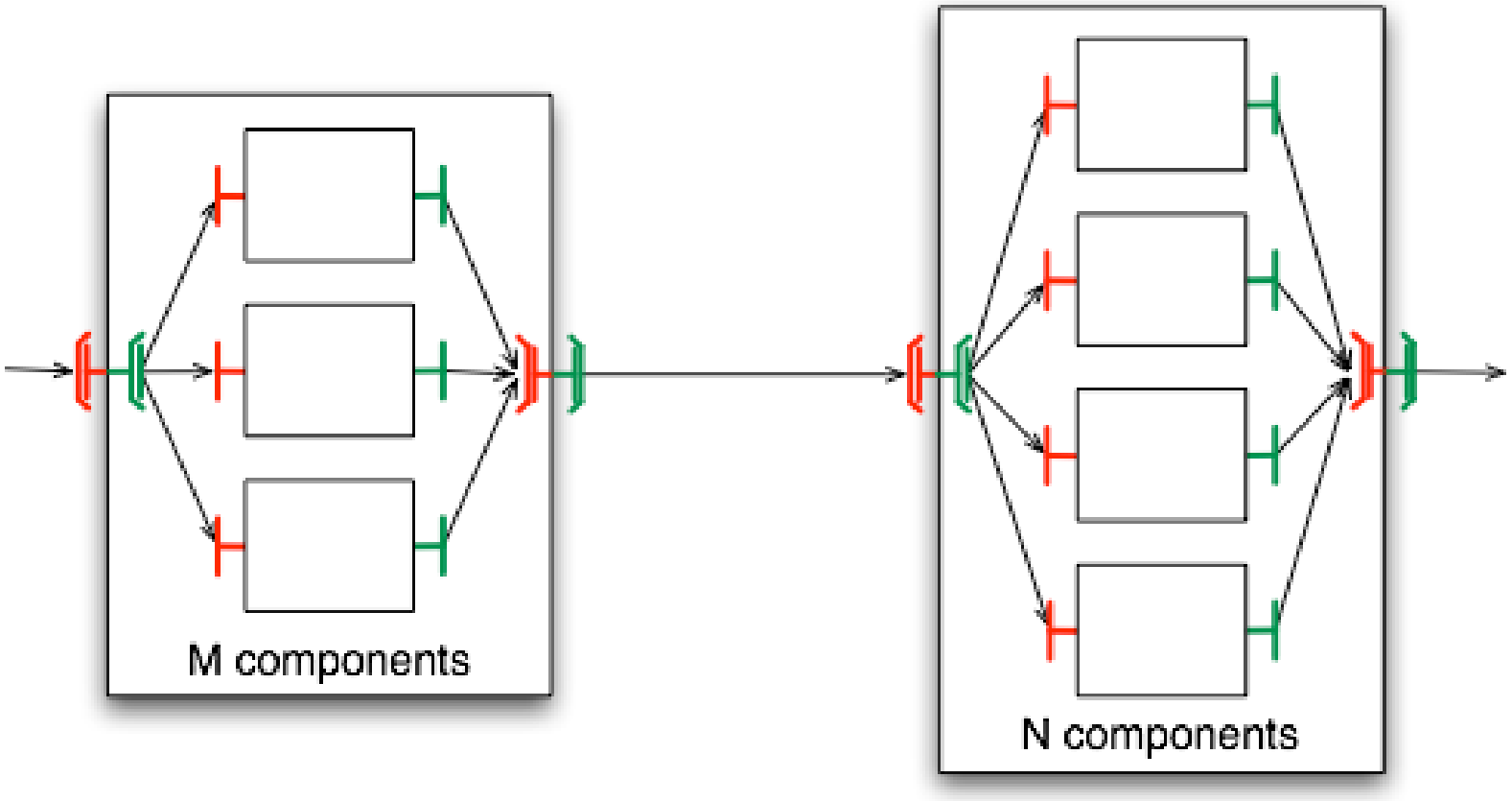
```
InterfaceType createFcItfType (
    …
    string cardinality
)…
```



- `CollectiveInterfacesController`
  - Collective interface policy
  - On a per-interface basis
  - Specified at instantiation-time
  - May be updated dynamically

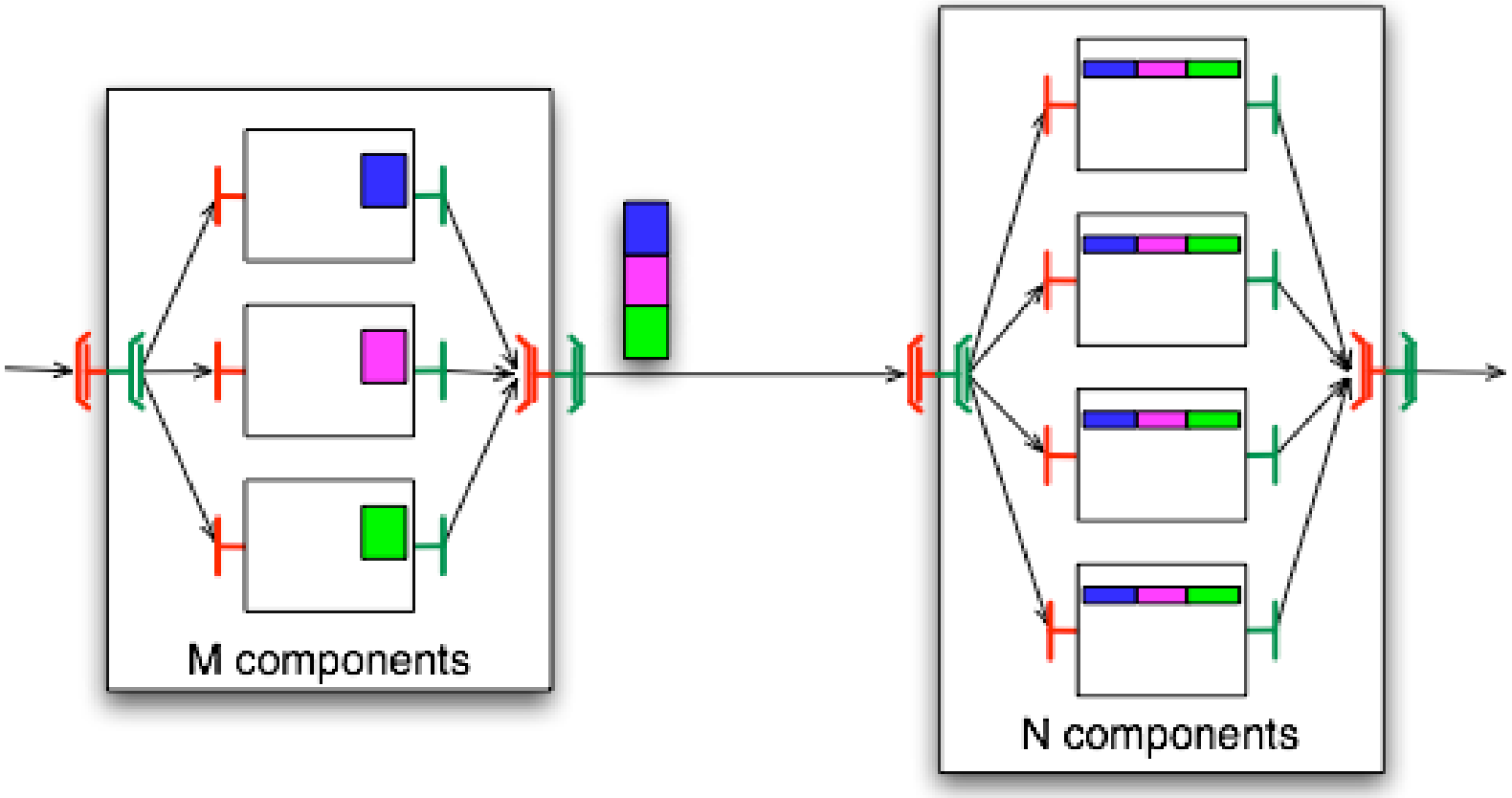
Perspective: MxN communications

- The problem



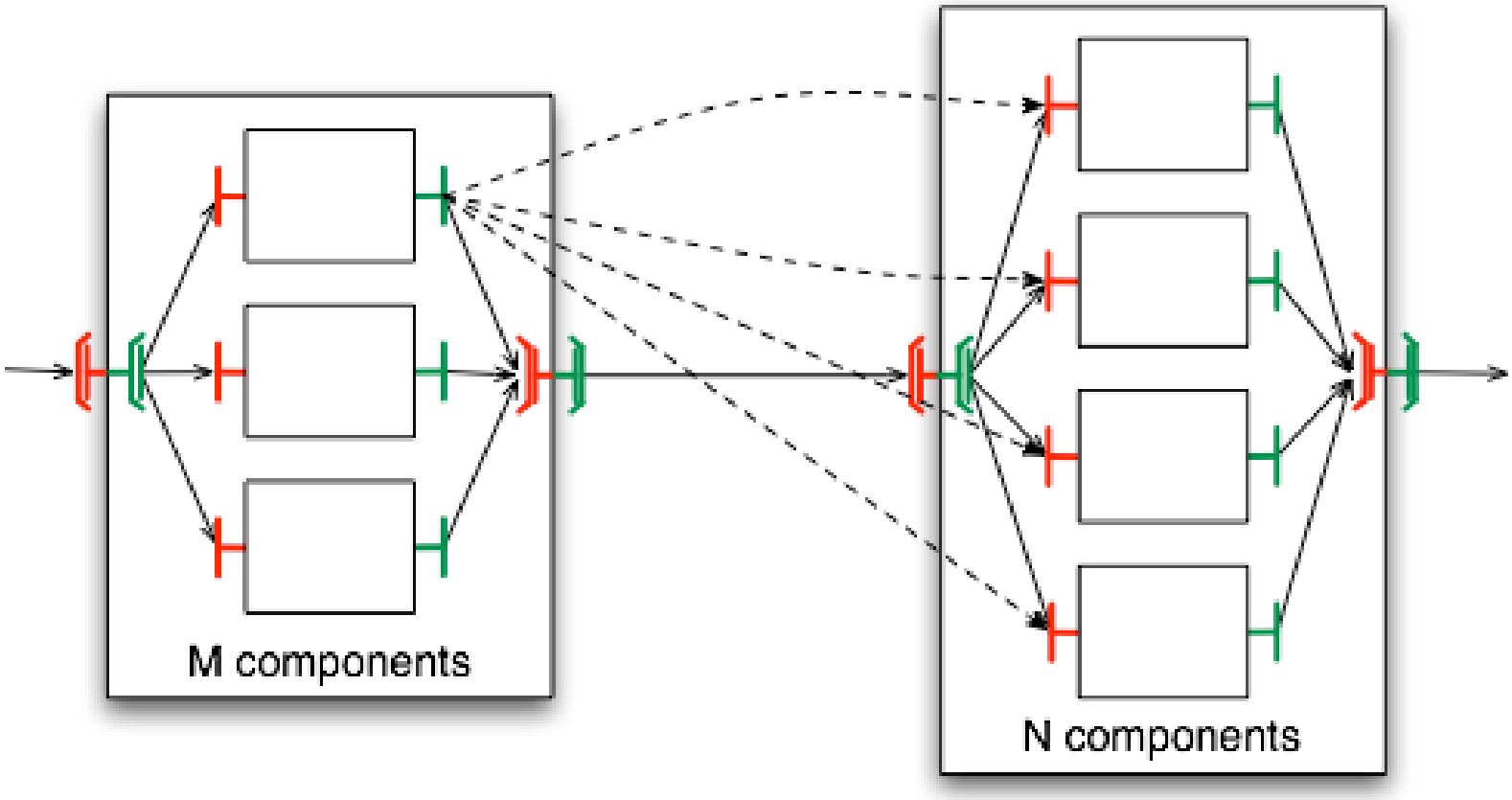
Perspective: MxN communications

- Data redistribution



Perspective: MxN communications

- Direct communications (⇔ tightly coupled applications)



Conclusion

- A proposal to address collective communications in Fractal
- Components expose the collective nature of their interfaces
- In order to simplify the programming model
- Collective communications handled by the framework
- Binding components not excluded
- Detailed specification available
- ProActive / Fractal will propose an implementation

Thank you !

Questions ?

Contact : matthieu.morel@sophia.inria.fr