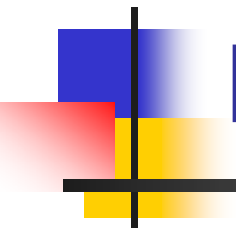
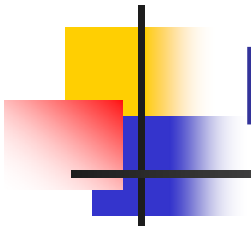


DREAM & Fractal

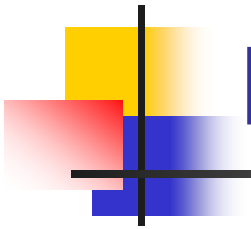
Vivien Quéma, Philippe Laumay
Projet Sardes, INRIA Rhône-Alpes

Fractal Workshop
January 29 2003



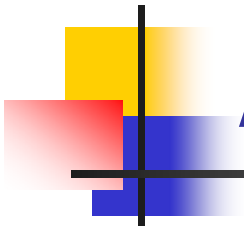
Plan

- Motivations
- DREAM overview
- Feedback



Motivations

- One objective of *Sardes* is to provide administration functions for scalable distributed systems
- The primary administration task is the system observation (monitoring)
  - Requirements of scalable observation functions
- Observation ?
  - Need to reify system state changes
    - ➔ What to monitor ? How to monitor ? …
  - Need of an infrastructure to send and receive events
    - ➔ ***Open technology for dynamic creation of events channel***

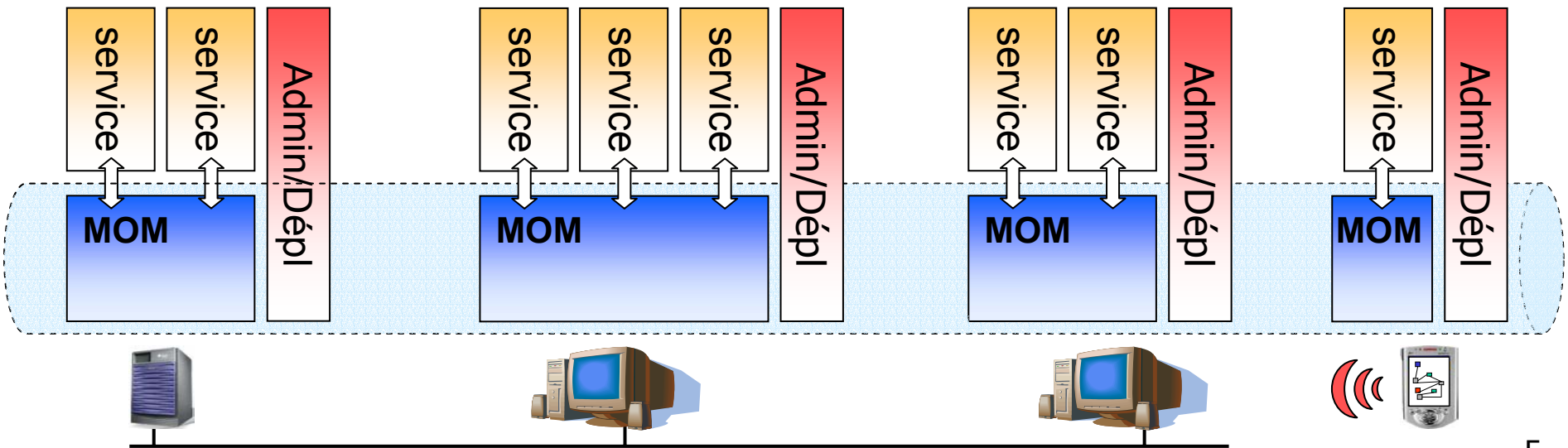


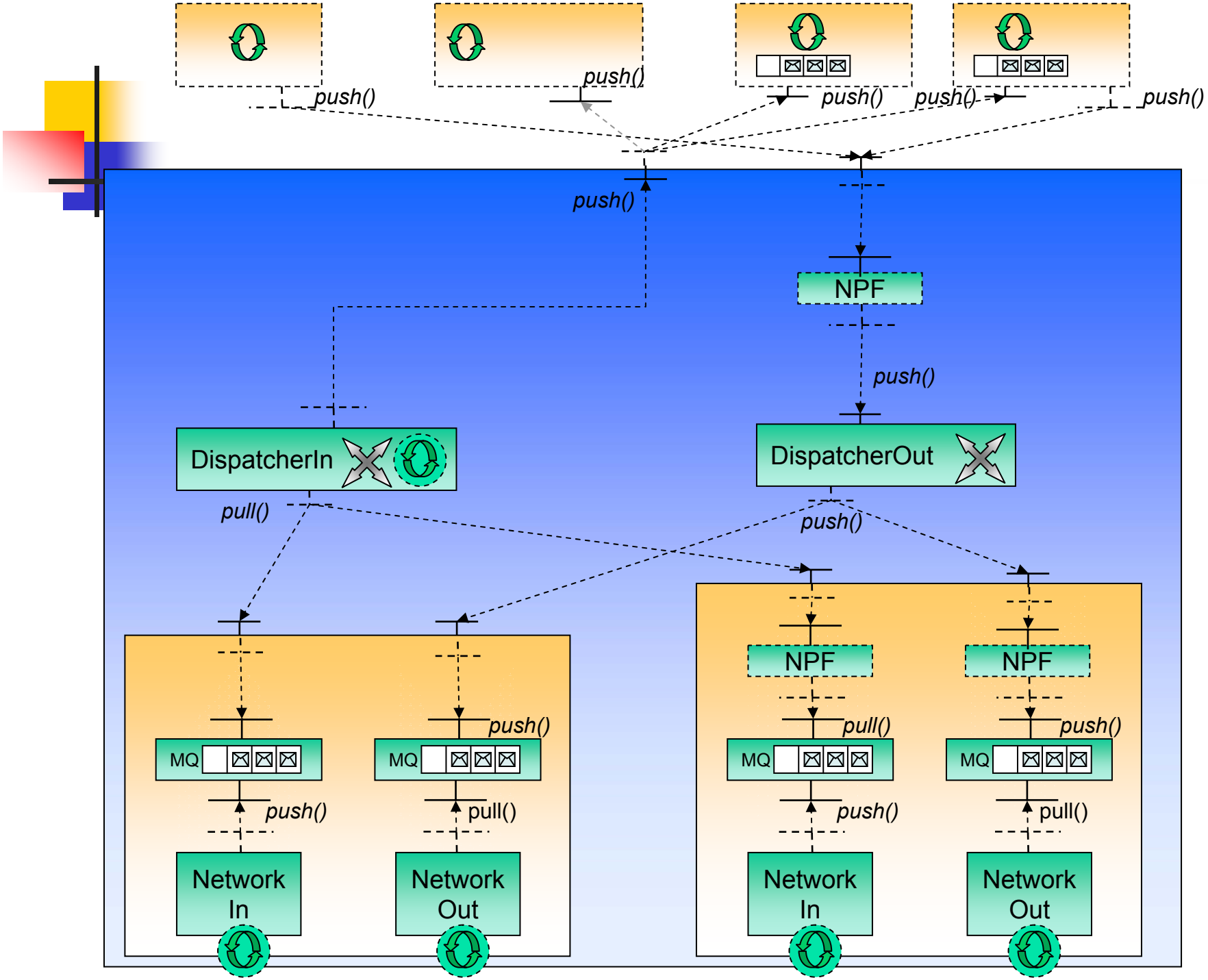
Adaptive asynchronous middleware

- Objectives
  - Asynchronous
  - Modular (component-based)
  - Configurable
  - Adaptive (dynamically reconfigurable)

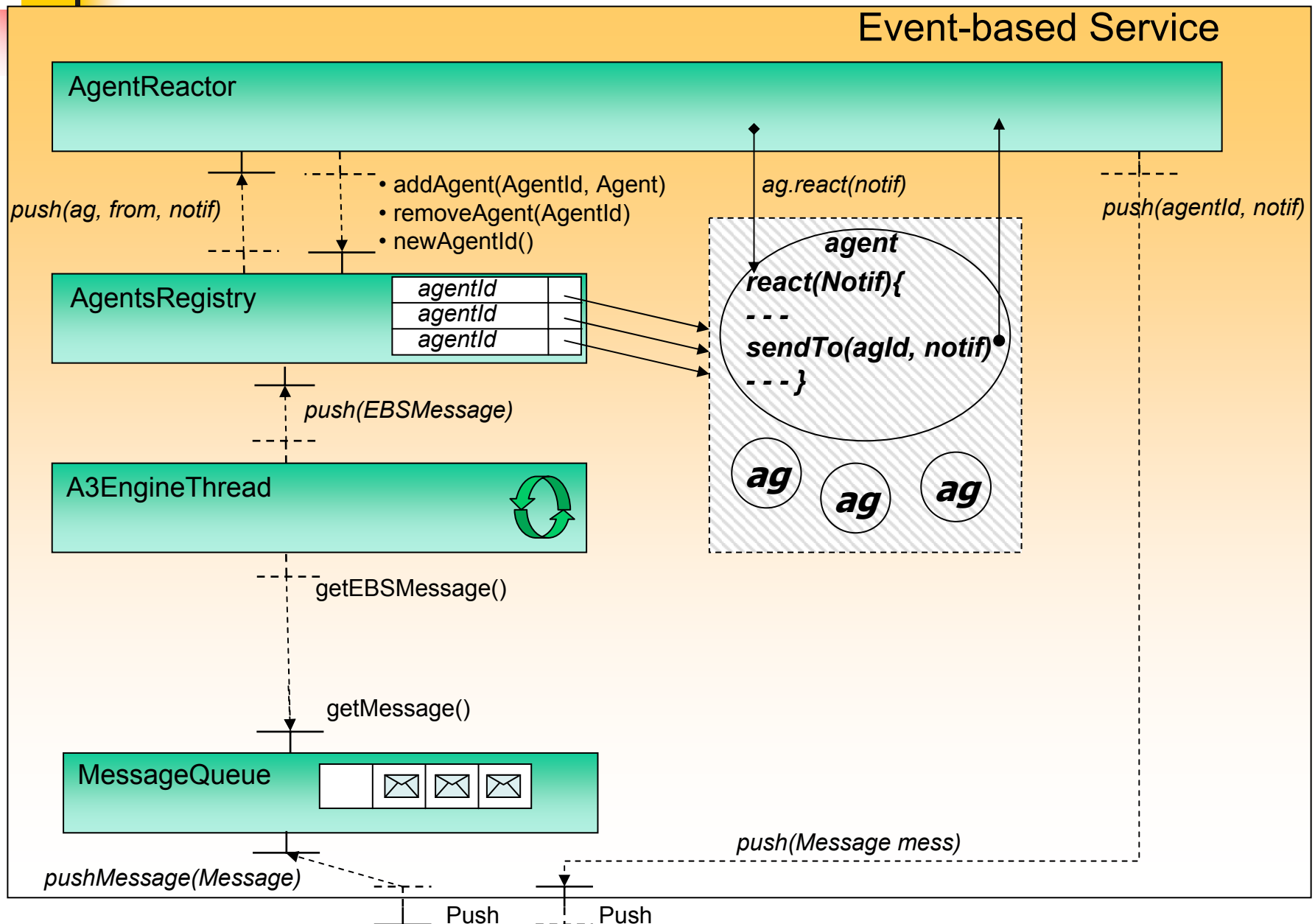
DREAM : a Dynamic REflective Asynchronous Middleware

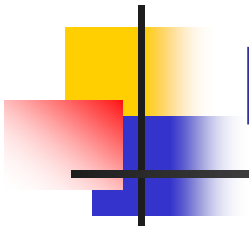
- Message-Oriented Middleware
  - Provides message queuing facilities
- Services use the MOM
  - Produce and consume messages
  - Implement a specific functionality
- Deployment and administration tool on each site




push()
push()
push()
push()
push()
push()
NPF
push()
DispatcherIn
DispatcherOut
pull()
push()
NPF
NPF
MQ
push()
pull()
MQ
push()
MQ
pull()
MQ
push()
pull()
push()
pull()
Network In
Network Out
Network In
Network Out

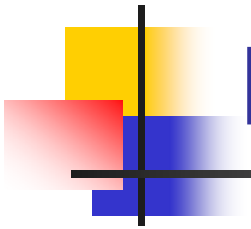
Event-based service





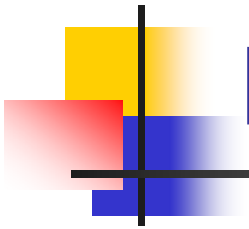
Feedback (1)

- This preliminary work had two objectives
  - Componentization of the A3 middleware
  - Learn the fractal framework and evaluate the advantages of fractal over traditional object languages and over other component models



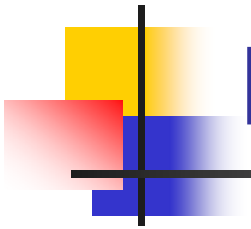
Feedback (1)

- Fractal is a minimal component model (not a framework dedicated to "business components")
  - Adapted to the construction of middleware
    - Non functional services are not provided
    - Control interfaces are not imposed by the framework
      - It is possible to define our own controllers to implement required control operations
    - It is possible to control the overhead induced by the Julia implementation thanks to code optimization



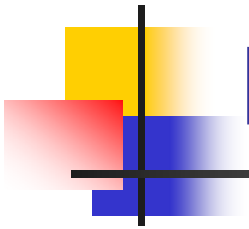
Feedback (2)

- Fractal
  - Allows to hierarchically structure the middleware
    - Primitive components for basic functions
      - Message queues
      - Sockets
      - Dispatchers
      - …
    - Composite components
      - Composing several components to form higher-level components
        - MOM, Services
        - Network



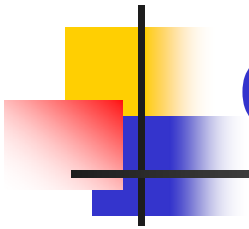
Feedback (3)

- Fractal is recursive
  - Composite may encapsulate other composite
    - Network composites are encapsulated in the MOM composite
  - That allows a hierarchical distribution of control operations
    - Each composite only manages the components it encapsulates
    - For example, 2 network composites may have different reconfiguration capabilities



Future work

- Study dynamic aspects of the MOM reconfiguration
  - Definition of other control interfaces ?
  - Specialization of existing controllers



Conclusion

- Questions ?