Component – Container – Connector Middlewares

(for Dynamic Reconfiguration support)

Frédéric Loiret, Ansgar Radermacher, François Terrier
CEA-LIST / L-LSP / Accord Team

Application domains concerned

- **« Strongly constrained »**
  - No dynamic instanciation during application lifecycle
  - All is envisaged off-line, statically at design time
  - Reconfiguration = operational modes
  - e.g. OSEK based implementations

- **« Less constrained »**
  - SW architecture evolves during application LC
  - Dynamic reconfiguration needs
  - « Sporadic » QoS needs

Application-level : Software Components

- **Objectives**

  - Isolate application functions design from the platform specificities
    - ✓ Notions of Middleware and Operating Systems
  - Declarative definition of non-functional aspects (QoS, tasks…)
  - Introduce capabilities to build a system through composition of exchangeable parts
    - ✓ Notions of component
  - Declare QoS on components interfaces

Middleware for RT/E Systems

- **Motivations**
  - ➢ SW part in RT/E Systems is increasing
  - ➢ SW in RT/E Systems is becoming more complex
    - ✓ More connectivity inside the system and with external world
    - ✓ Dynamic adaptation of RT/E System operation

- **Constraints**
  - ➢ QoS issues are critical (costs, quality)
    - ✓ Performances, resource consumption, satefy/security
  - ➢ Platforms are heterogeneous and variable
  - ➢ Unsynchronised evolutions of HW and SW

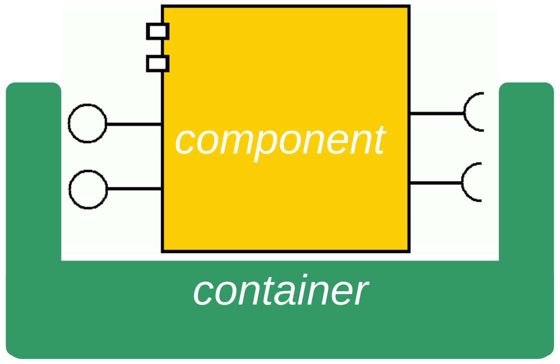
Container/Component model

- **Component alone are not sufficient**
  - ☺ Ease design at application level by assembly of parts
  - ☹ *Dependency to platform is located inside the component, but remains explicit in the code of the application functions*
  - ☹ *QoS declaration is only informative (comments)*

- **Middleware alone are not sufficient**
  - ☺ Introduces a standard view of platform specificities
  - ☹ *Moves dependencies from platform to middleware*
  - ☹ *Introduces systematic overheads*

➔ Container/component oriented Middleware for RT/E Systems

Container

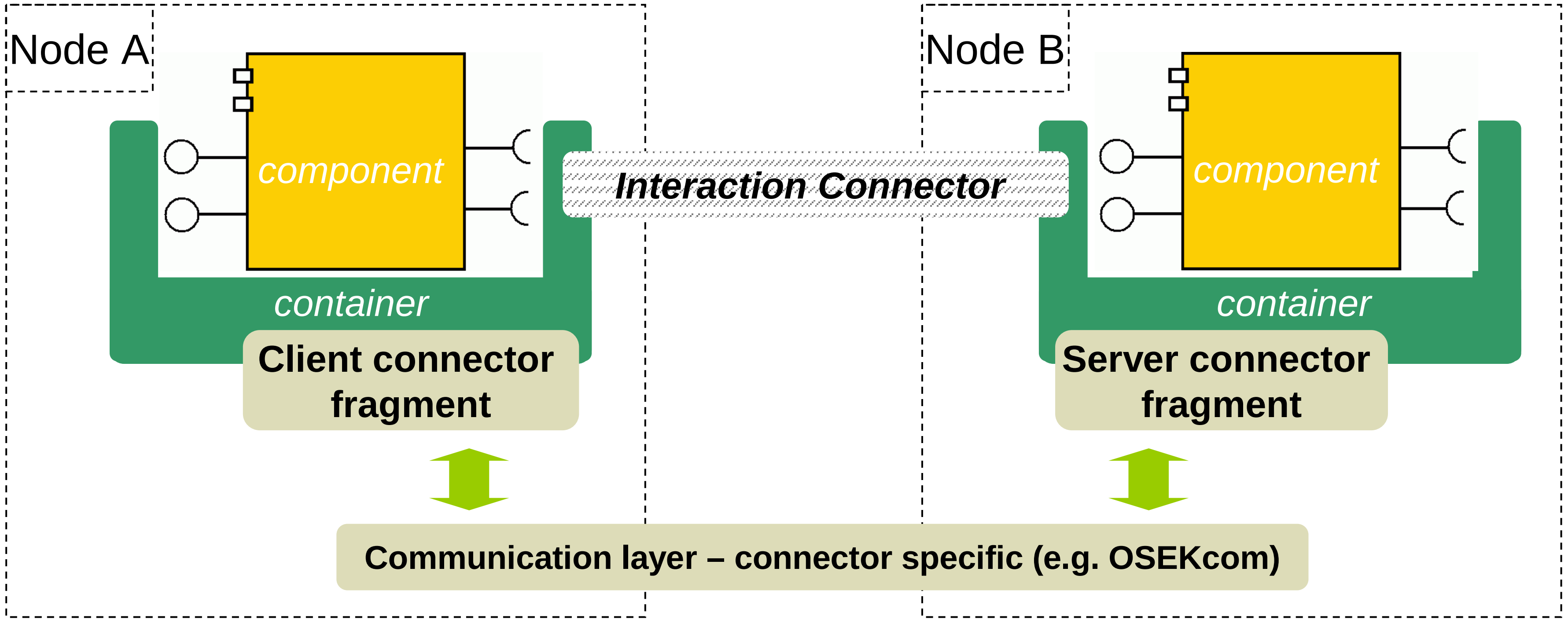


- A **generated** wrapper dedicated to functional components
- Provides the glue between component and its environment
- A decoupling of non-functional aspects and application functional logic
- Implements arbitrary non-functional aspects
  - ✓ Connectors
    - → Communication protocols and synchronization mechanisms
  - ✓ Task allocation
  - ✓ Fault tolerance
  - ✓ **Reconfiguration management**
  - ✓ **…**
- Embedes only required non-functional services

eMC^3 : Embedded Middelware based on Component-Container-Connector

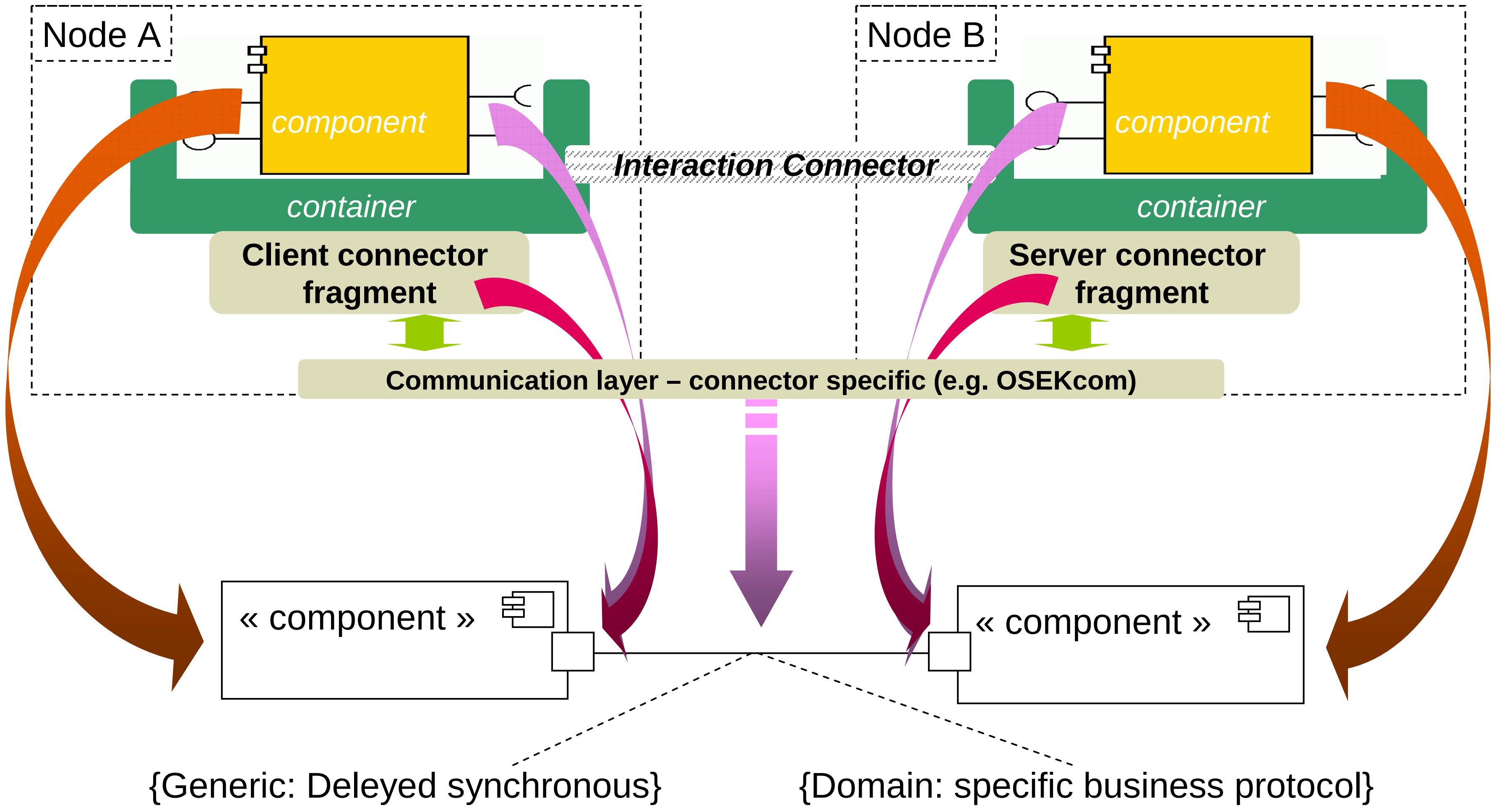
eMC³ : Introducing connectors

- **Software entity managing inter-components interaction:**
  - May be considered as part of the container
  - Fragmented
  - Communication layer specific to the connector
  - (potentially) complex intermediary processing


Node A
component
container
Client connector fragment
Interaction Connector
Node B
component
container
Server connector fragment
Communication layer – connector specific (e.g. OSEKcom)

From models to implementations

- **Conceptual mapping with UML components**


Node A
component
container
Client connector fragment
Interaction Connector
Node B
component
container
Server connector fragment
Communication layer – connector specific (e.g. OSEKcom)
« component »
« component »
{Generic: Deleyed synchronous}
{Domain: specific business protocol}

Orientations : REVE project

- Ongoing work

- Build a component model and a runtime
- where policies for handling context changes can be specified and programmed
- Context changes depending on QoS properties
  - **Application dependent properties**
  - **Resources : Memory, CPU, Network**

- Based on Fractal/Think
  - **A component-based framework dedicated to operating system design**

- Perspective : a container-oriented approach to manage dynamic reconfiguration
  - **Based on Qinna Framework**