

# Adding the *“Webtone for Information”*

*(“Dealing with real-time in OPENWINGS”)*



# CONTENTS

- Sharing the vision
- Strengthen the vision
- Implementing the joint vision
- Exploiting a combined vision



# SHARING THE VISION

*(“MATCHING REQUIREMENTS”)*



## SHARING THE VISION: global trends

*(Swiss Federal Institute of Technology (ETH) Zurich, August-2000)*

“In the near future, distributed application frameworks will support **mobile code, multimedia data streams, user and device mobility, and spontaneous networking**. **Scalability, quality of service, and robustness with respect to partial component failures** will become key issues.”

“The overall trend in communication paradigms seems to be clear: It goes from the simple procedural paradigm requiring relatively little system support, via the object-oriented method invocation principle, towards more abstract schemes for **loosely coupled asynchronous systems** that require **complex run-time infrastructures**. **Scalability** of such infrastructures is a real challenge, however.”

“Furthermore, **CORBA** was conceived for static distributed systems, requires considerable resources at run time, and uses the **traditional client-server** model as the basic metaphor. It is therefore **not well suited** for small devices, highly dynamic systems, and services that are **spontaneously integrated into a federation**. This, however, is a major trend in distributed systems, to which Jini and similar systems are better adapted.”

*(<http://www.inf.ethz.ch/vs/publ/papers/DisSysUbiCompReport.html>)*



# SHARING THE VISION: Openwings

## (JavaOne '2000: T.O.C. requirements)

- Requirements
  - reduce a-priori configuration
  - flexible access to applications from various clients
  - addition of unplanned computing resources
  - applications developed as services available to all users (mobile-code, events, lease-model)
  - infrastructure needs to allow for the fact that resources are changing
  - services & clients must be able to adapt to the removal or addition of services (nomadic users)
- Functional benefits
  - reduced (little or no) administration
  - reduced development
  - increased mobility on heterogeneous platforms
- Service oriented architecture characteristics
  - services run as either persistent ('activatable' / 'always-active') or transient ('on request') services
  - all services are administratable, configurable & provide logging & tracking
  - *“easy”, “dynamic”, “de-coupled” & “adaptive”*



# SHARING THE VISION: System Overview

- **Sensors**
  - all networked devices
  - 4.000 Hz. updates

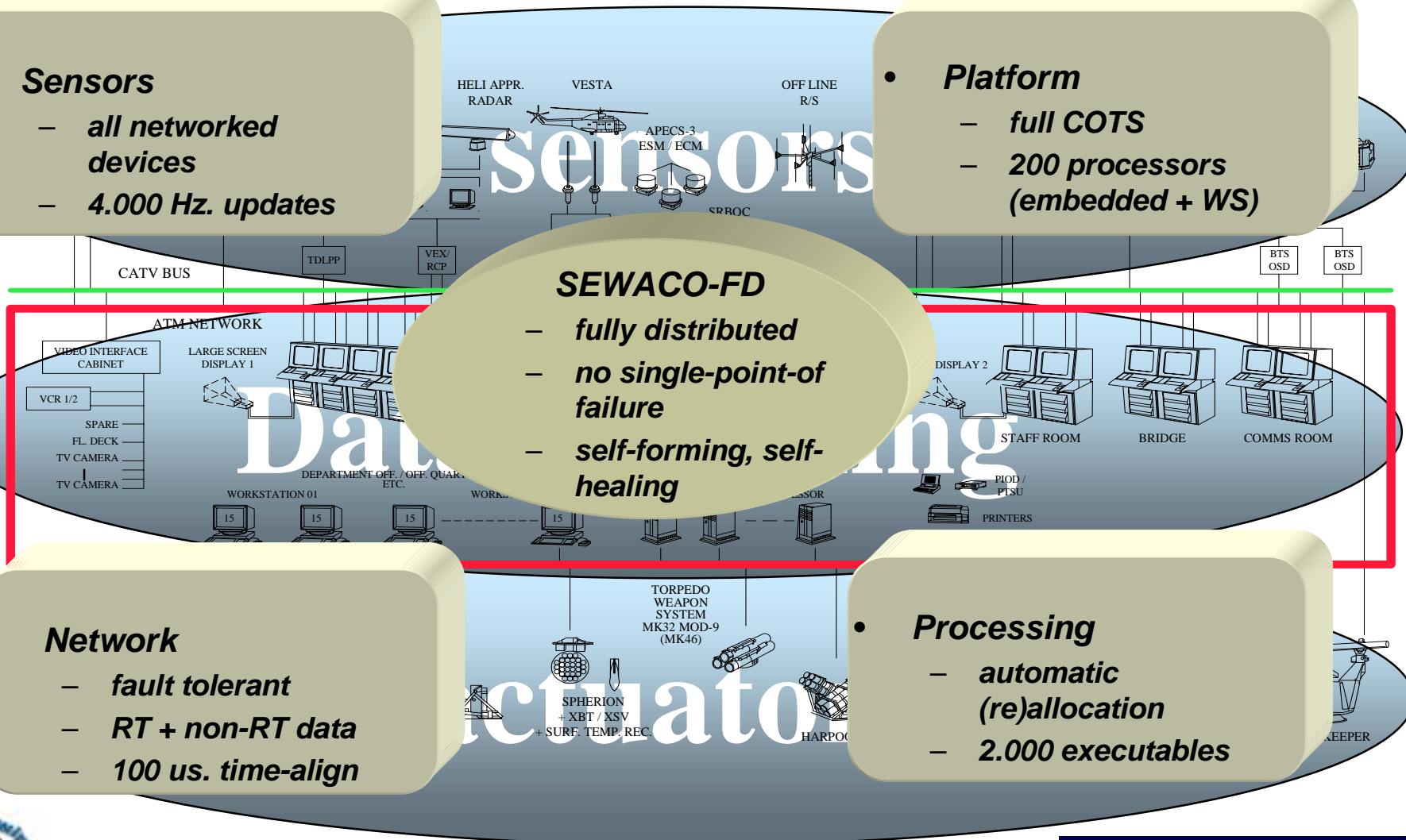
- **Platform**
  - full COTS
  - 200 processors (embedded + WS)

## SEWACO-FD

- fully distributed
- no single-point-of failure
- self-forming, self-healing

- **Network**
  - fault tolerant
  - RT + non-RT data
  - 100 us. time-align

- **Processing**
  - automatic (re)allocation
  - 2.000 executables



# SHARING THE VISION: SEWACO-FD

(Signaal '92: required architectural transparencies)

## Transparency

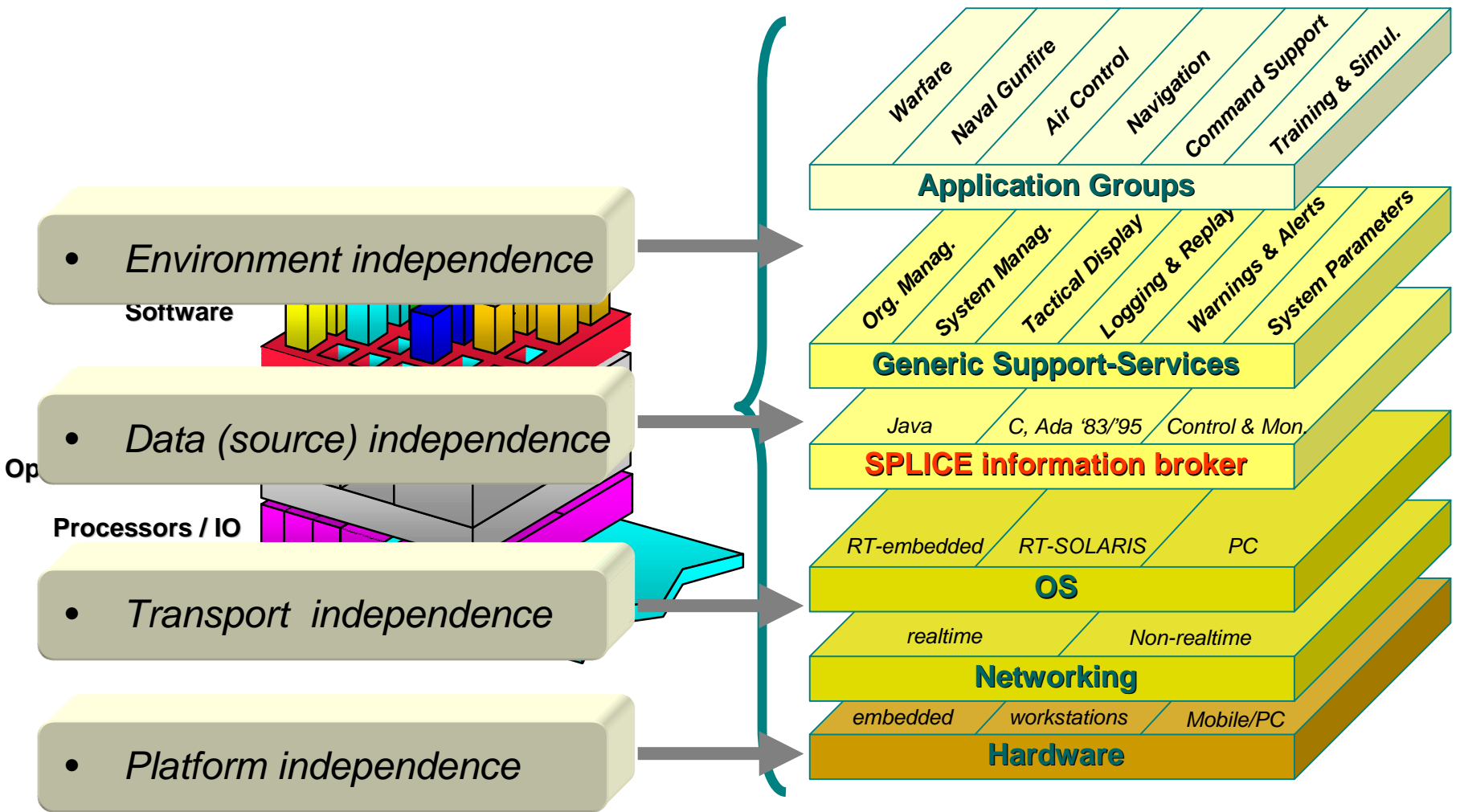
## Motivation

- COTS support
  - *technology* - : *decoupling* of applications from underlying technology
- Real-time support
  - *frequency* - : *decoupling* of produced/consumed data-rates
  - *timing* - : *decoupling* of information-accuracy and -latency
- Distribution support
  - *access* - : *identical* operations for local & remote applications
  - *location* - : *transparent* communication, no location knowledge
- Fault-tolerance support
  - *replication* - : *transparent* replication to create redundancy
  - *failure* - : *automatic* detection & recovery of system failures
  - *migration* - : *automatic* re-allocation of (failed) application programs
- Flexibility & lifecycle support
  - *performance* - : *runtime* reconfiguration to handle load scenario's
  - *scaling* - : *lifetime* re-use of application SW with expanding scale



# SHARING THE VISION: SEWACO-FD

(layered architecture, Openwings 'independencies' similarities)



# STRENGTHEN THE VISION

(“adding the: *web-tone for information*”)



# STRENGTHEN THE VISION

(adding 'information')

- Service-centric

- any service
- any where
- any time

- Information-centric

- right information
- right place
- right time

JINI

- **'Webtone' for services**

- spontaneous
- self-healing

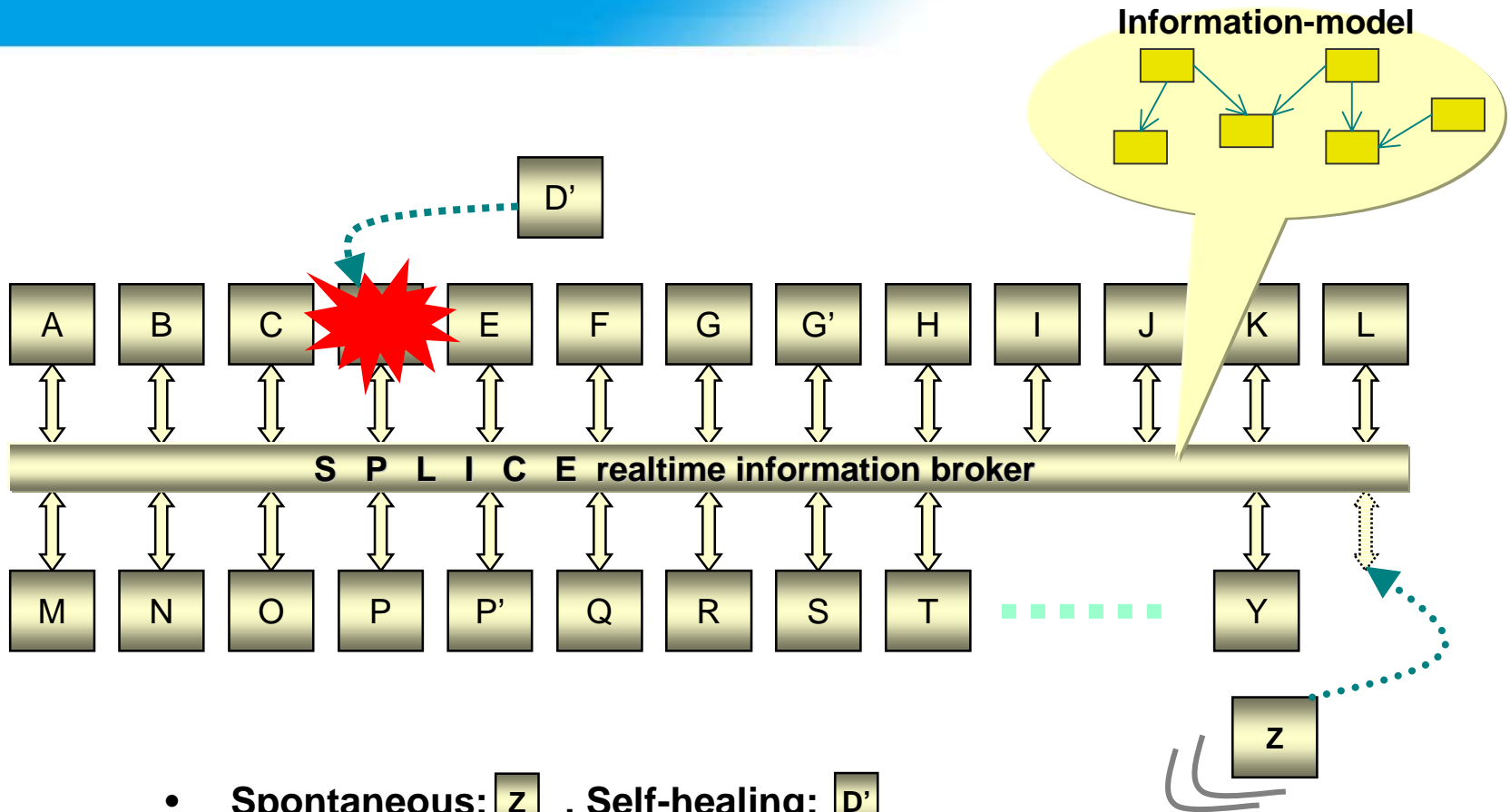
SPLICE

- **'Webtone' for information**

- autonomous
- real-time



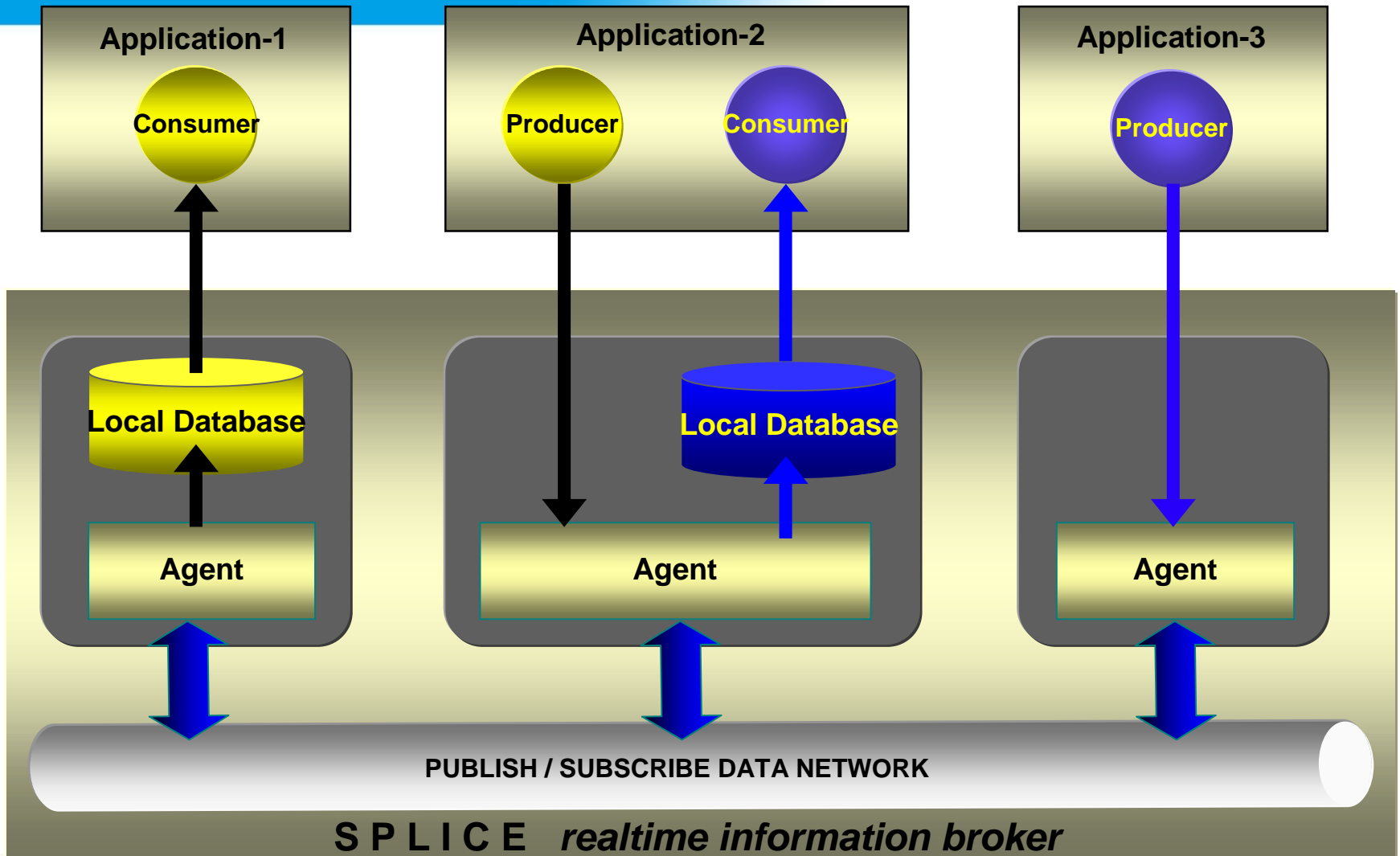
# SPLICE: INFORMATION BACKBONE



- Spontaneous: **Z** , Self-healing: **D'**
- Redundant & Replicated: **G'**, **P'**
- Implicit, shared and guaranteed backbone quality: **S P L I C E**  
(peer-to-Peer QoS: delivery, persistency, priority)



# SPLICE: ARCHITECTURE

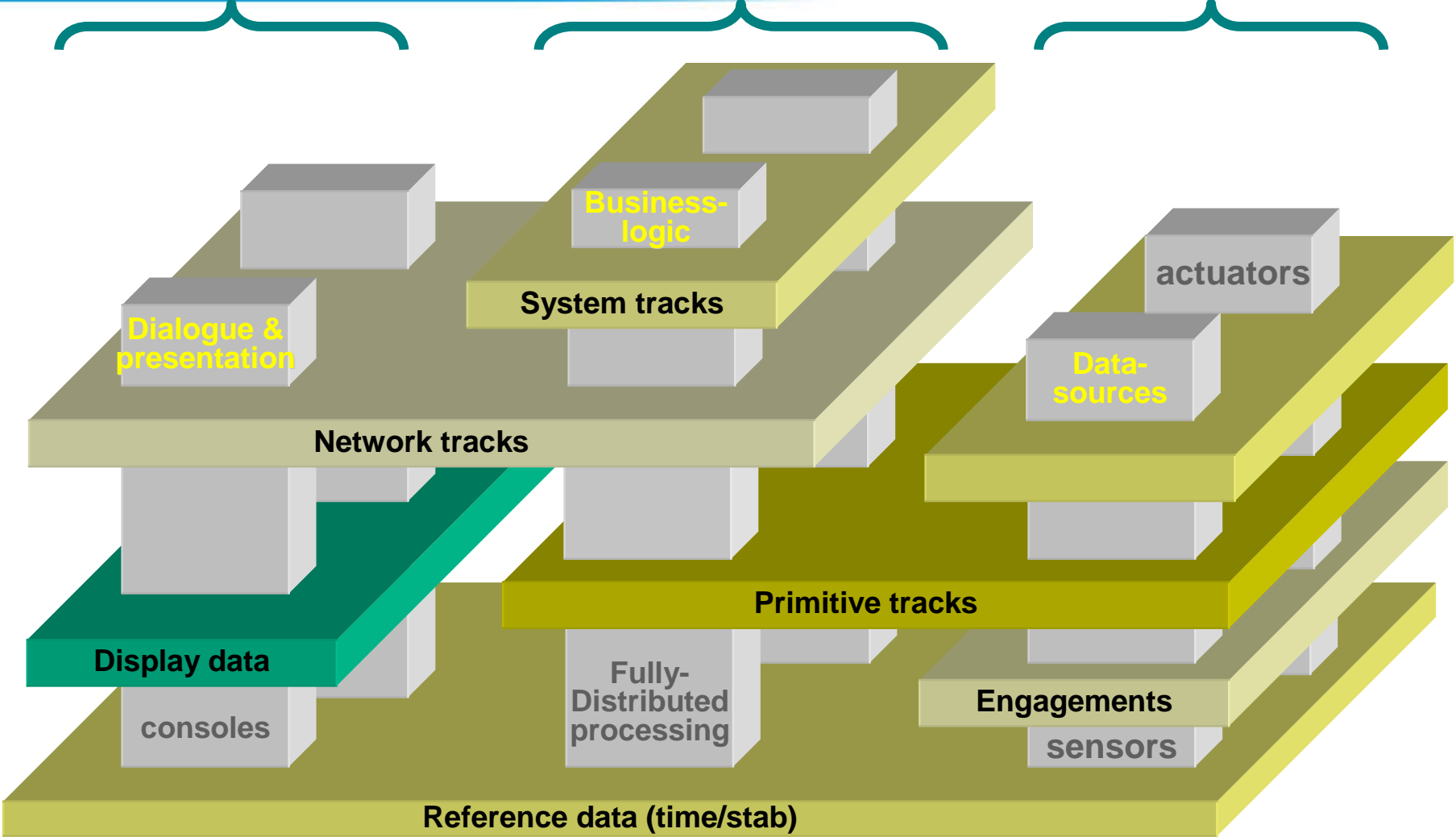


# SPLICE: Self-forming data-planes & 3-tier processing

Near-Realtime HCI

Realtime Command & Control

Hard-realtime sensor/weapon IO

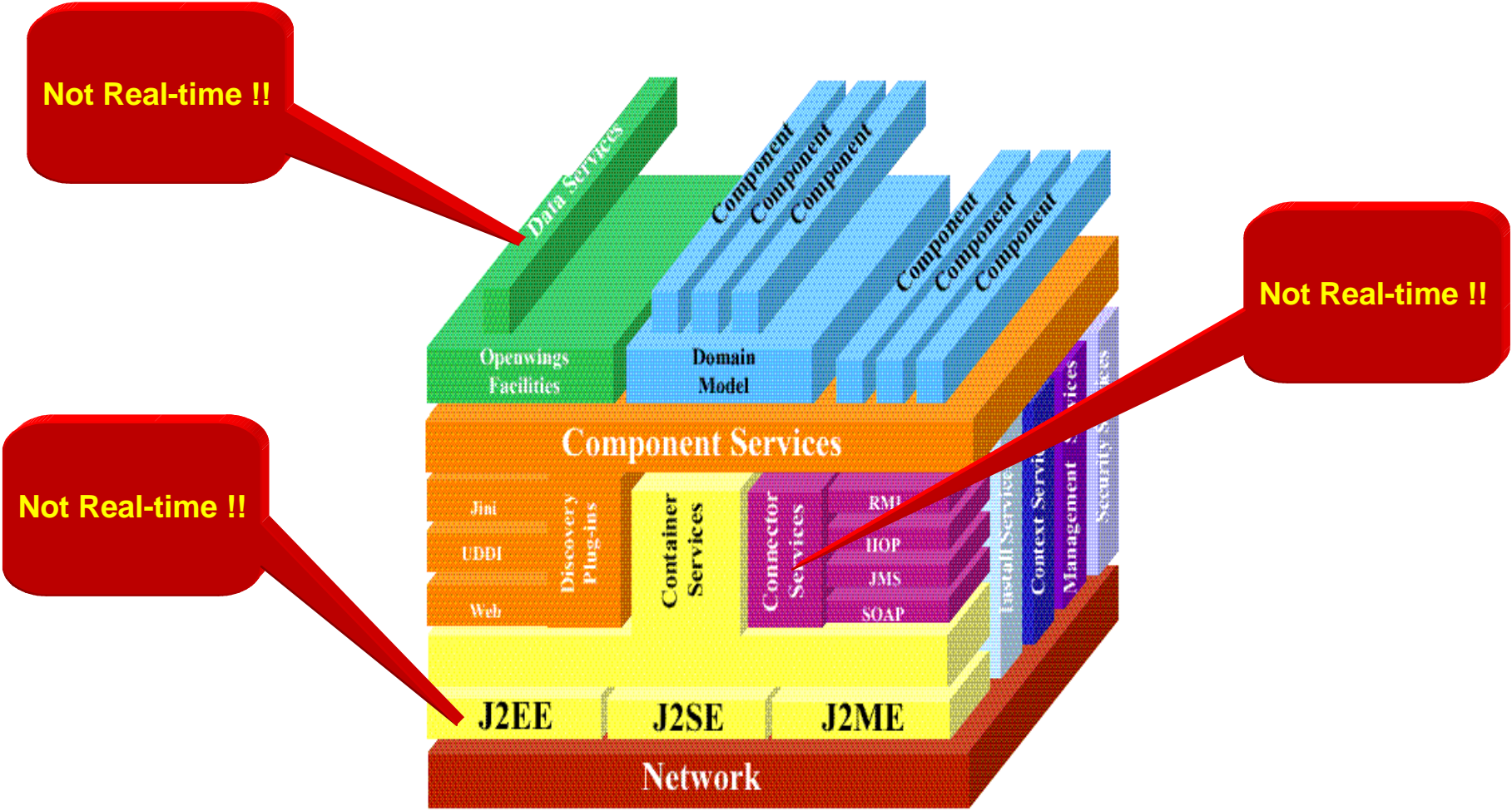


# IMPLEMENTING THE JOINT VISION

*(“adding real-time data to Openwings”)*

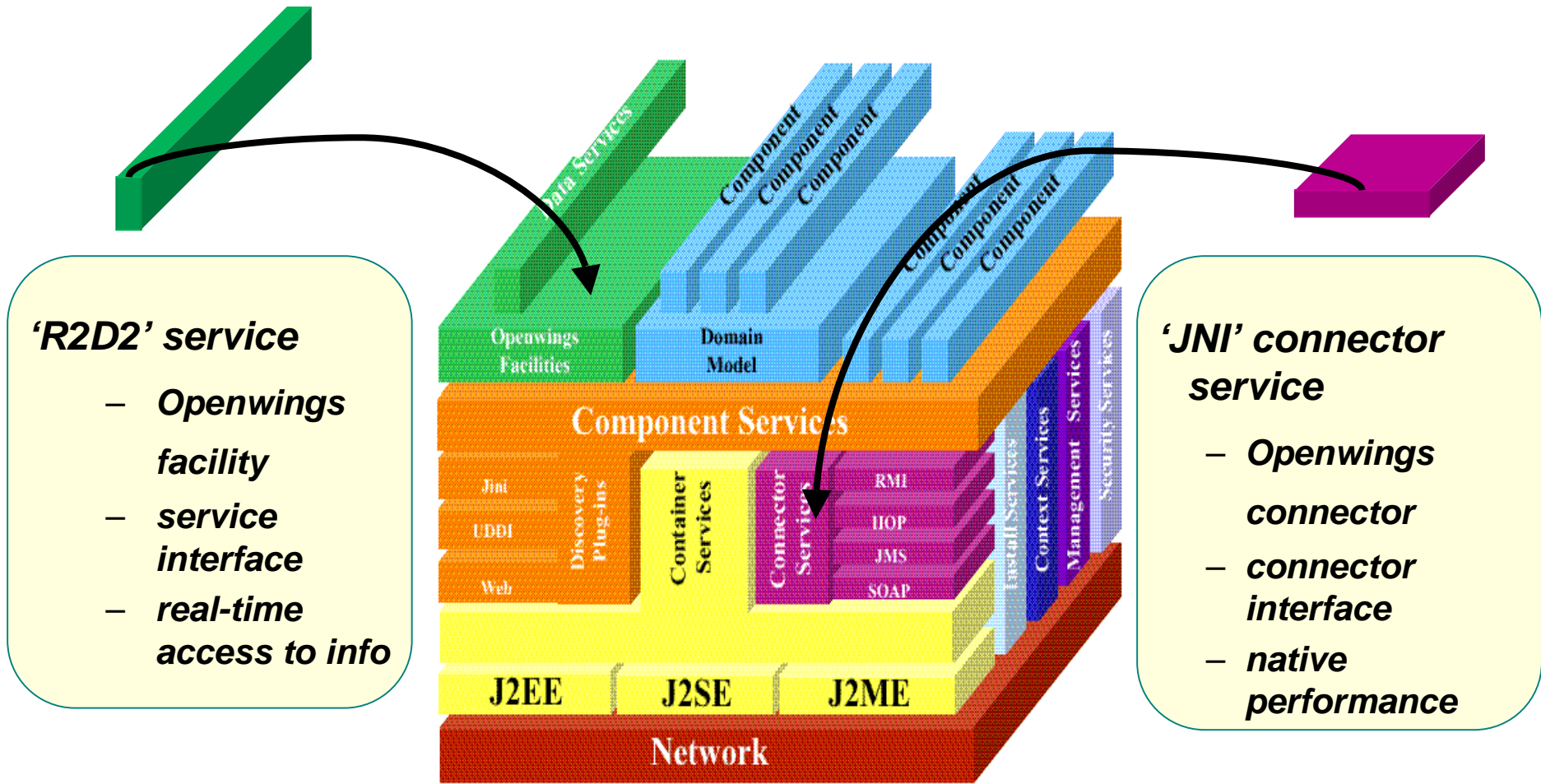


# IMPLEMENTING THE JOINT VISION (real-time requirements)



# IMPLEMENTING THE JOINT VISION

*(integrating the Realtime Replicated Distributed Dataspaces 'R2D2' service)*



### 'R2D2' service

- *Openwings facility*
- *service interface*
- *real-time access to info*

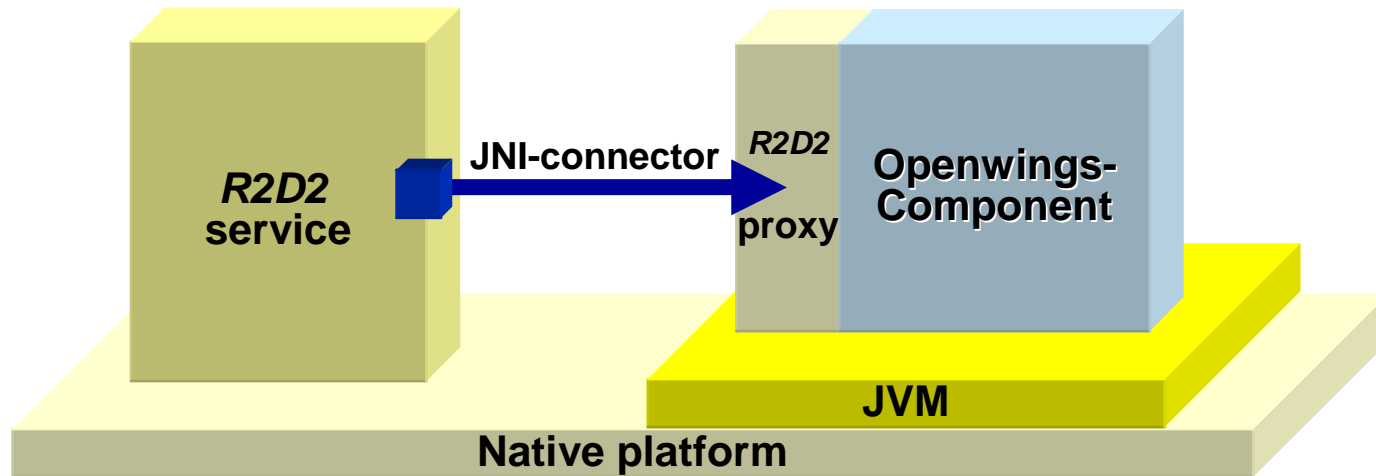
### 'JNI' connector service

- *Openwings connector*
- *connector interface*
- *native performance*



# IMPLEMENTING THE JOINT VISION

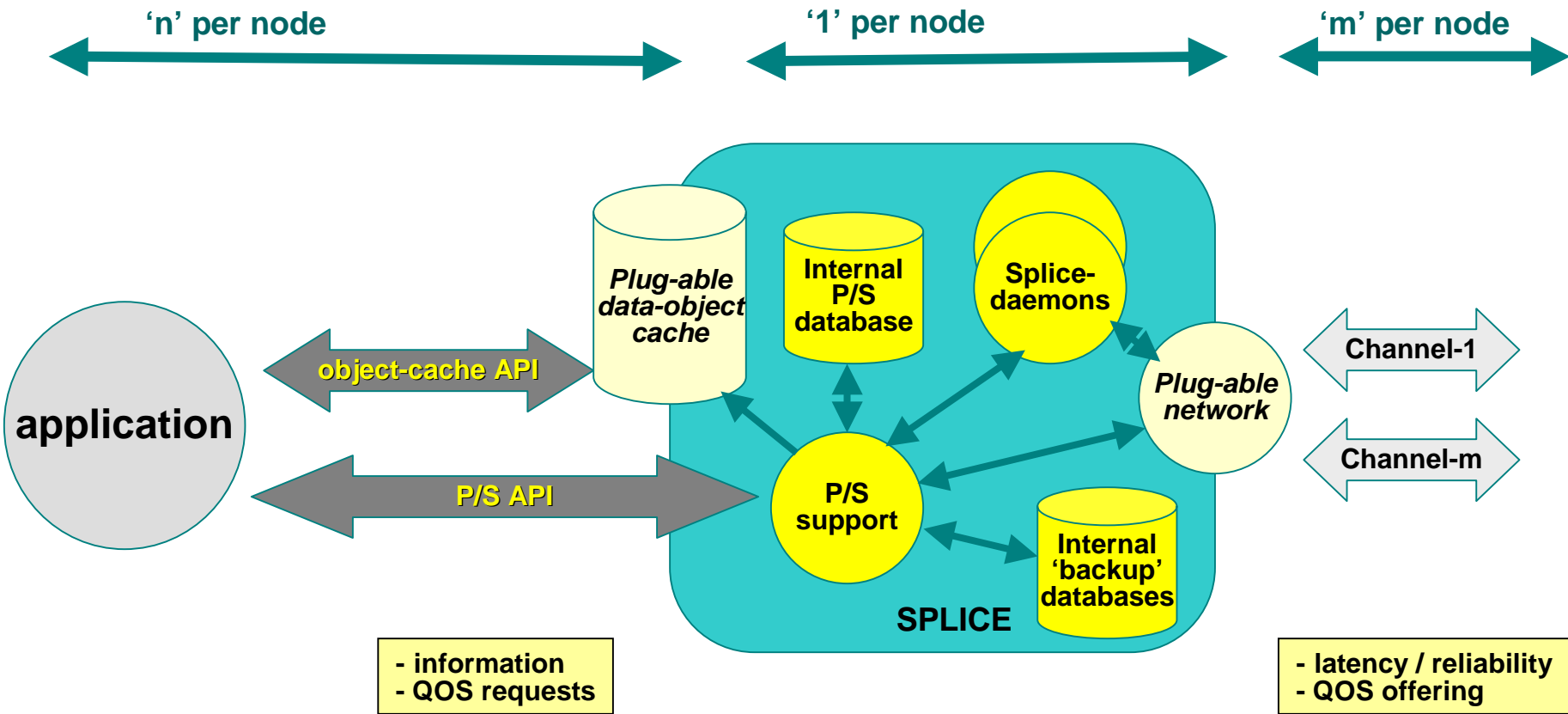
(proposing the 'R2D2' service)



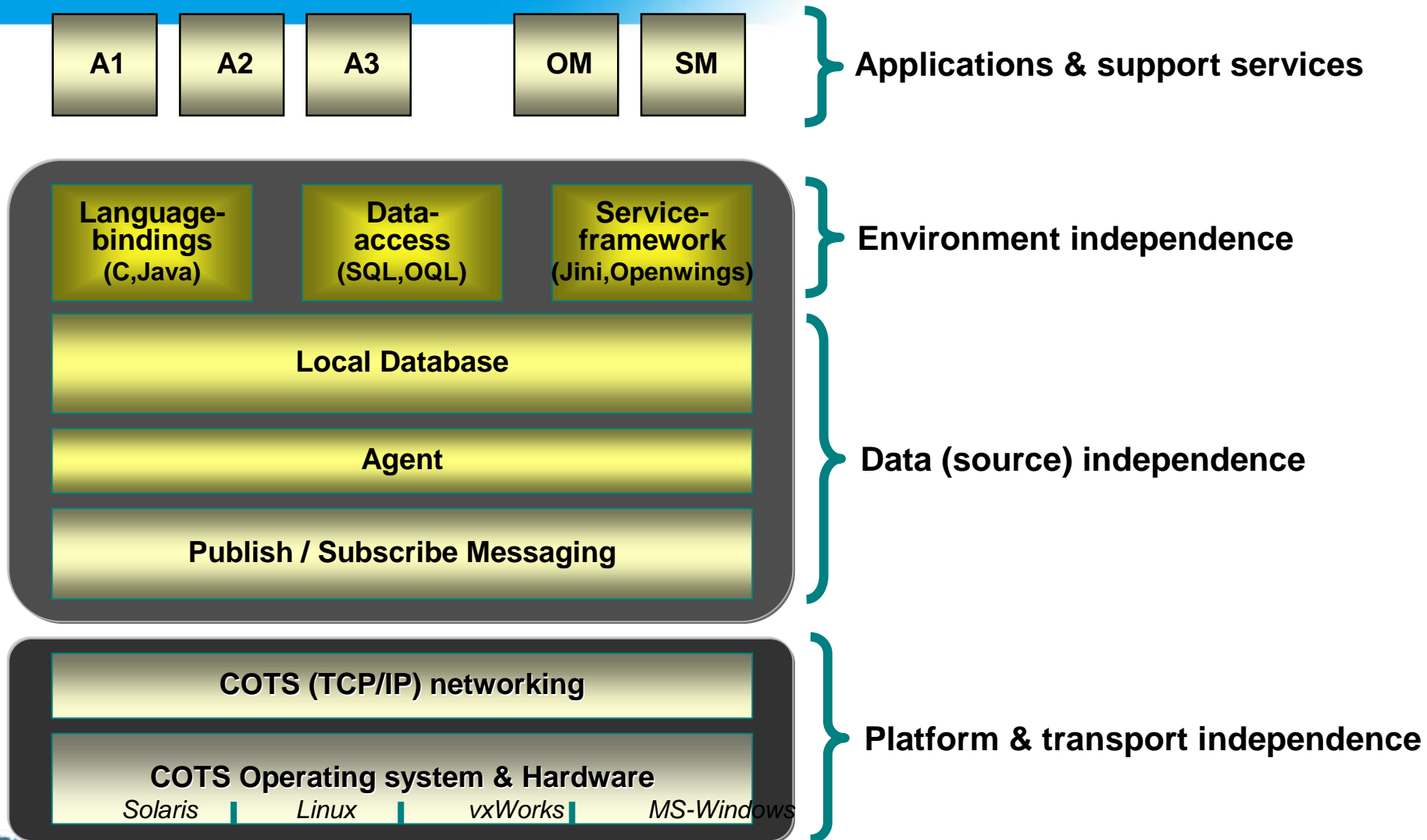
- *R2D2* = Realtime Replicated Distributed Data-spaces
  - **Real-time** : *proper QoS (latency, reliability, persistency)*
  - **Replicated** : *selectively replicated so **locally-available** data*
  - **Distributed** : ***distributed information** for distributed components*
  - **Data-spaces** : ***local data-caches** with subscribed information*
- Specification & Implementation
  - **R2D2** : *Openwings facility-service **specification***
  - **SPLICE** : *Thales-NL high-performance native **implementation***



# IMPLEMENTING THE JOINT VISION (the SPLICE 'R2D2' service implementation)



# SPLICE: standards based & standards-compliant



# EXPLOITING A COMBINED VISION

*(“the right interface at the right time at the right place”)*



# COMBINED VISION: features & benefits

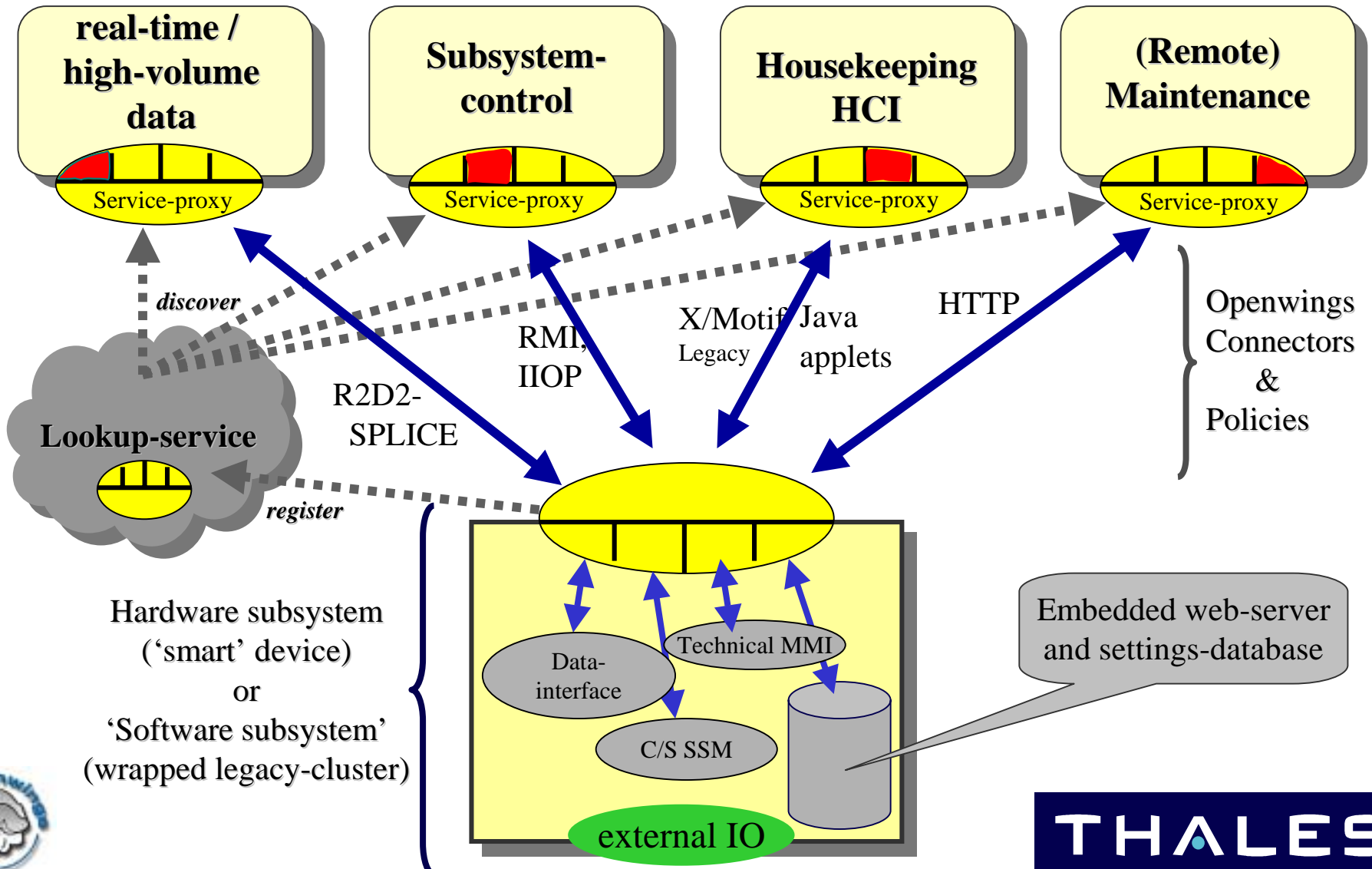
## (of an added 'information-centric' approach)

- System design
  - information model
  - state-based system
  - information classification
  - provides a **stable basis** to operate upon by applications
  - enhances component **autonomy**
  - allows transparent and global **QoS** assurance
  
- System development
  - minimized component dependencies
  - normalized information-system environment
  - simple concept
  - reduced **complexity** and enhanced re-usability
  - designed only once, offers shared/**guaranteed** properties
  - **small** learning effort and flat learning curve
  
- System integration
  - maximized component autonomy
  - globally accessible information
  - powerful information-broker concept
  - effortless component **integration**
  - **easy** monitor & control by availability of data + meta-data
  - radical **shift** in ratio between design and integration effort
  
- System deployment
  - realtime information broker (data-distribution)
  - global availability of application state (context)
  - distributed & fault-tolerant persistence
  - guaranteed QoS for **reliability**, **latency** and **persistence**
  - allows **runtime migration** of app's by retaining their state
  - allows applications to join the system at **any time**
  
- System maintenance & evolution
  - de-coupled components
  - global availability of all (time-stamped) data
  - highly adaptive and extensible information model
  - allows runtime replacement and **evolutionary upgrading**
  - extensive **support** for logging & replay of information
  - for a **future-proof, re-usable, robust and scalable** system



# COMBINED VISION: Service Interfaces

(data-, command-, HCI- and maintenance-interfaces & connectors)





# Questions ?

