PHILIPS

# Resource Reservation in Real-Time Operating Systems - a joint industrial and academic position

*Liesbeth Steffens, Philips Research*
*Gerhard Fohler, Mälardalen University*
*Giuseppe Lipari, Scuola Sup. S.Anna*
*Giorgio Buttazzo, Università di Pavia*

ARTIST International Collaboration Day 2003
*October 12, 2003*

# Reasons for resource reservation

- Temporal protection for system robustness
- Independent design, analysis, and validation of real-time subsystems
- Re-use of legacy applications
- Quality of Service (QoS)
- Hybrid open systems

# Application domains

- Aerospace
- Multi-media
- Real-time control systems

# Operating system trends

General purpose OS


Open
Multiple applications
Memory protection
Multiple units of failure

Timesharing

Temporally unpredictable
Large system overhead

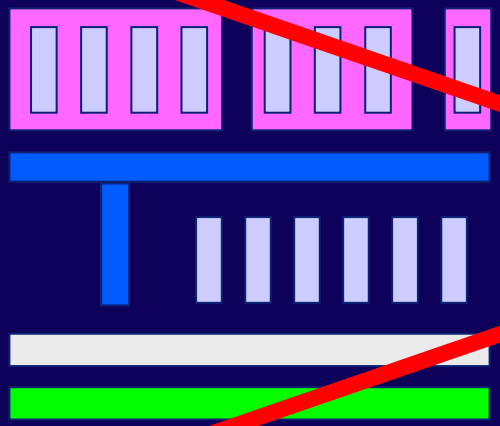
Closed
Single application
No memory protection
Single unit of failure

Priority scheduling

Temporally predictable
Small system overhead

Real-time OS

# 2 Approaches



processes

OS API

real-time tasks

RTOS API

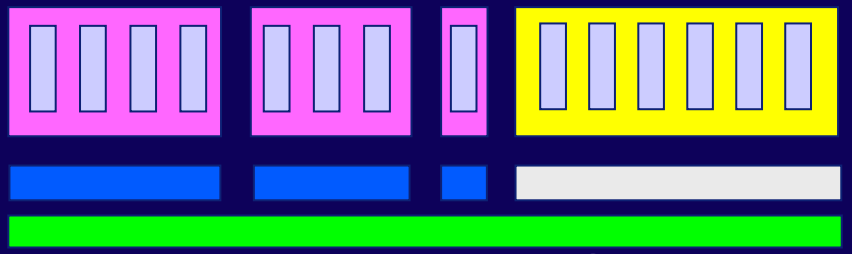
priority scheduling, lean, predictable

Approach 1

processes

(RT)OS API

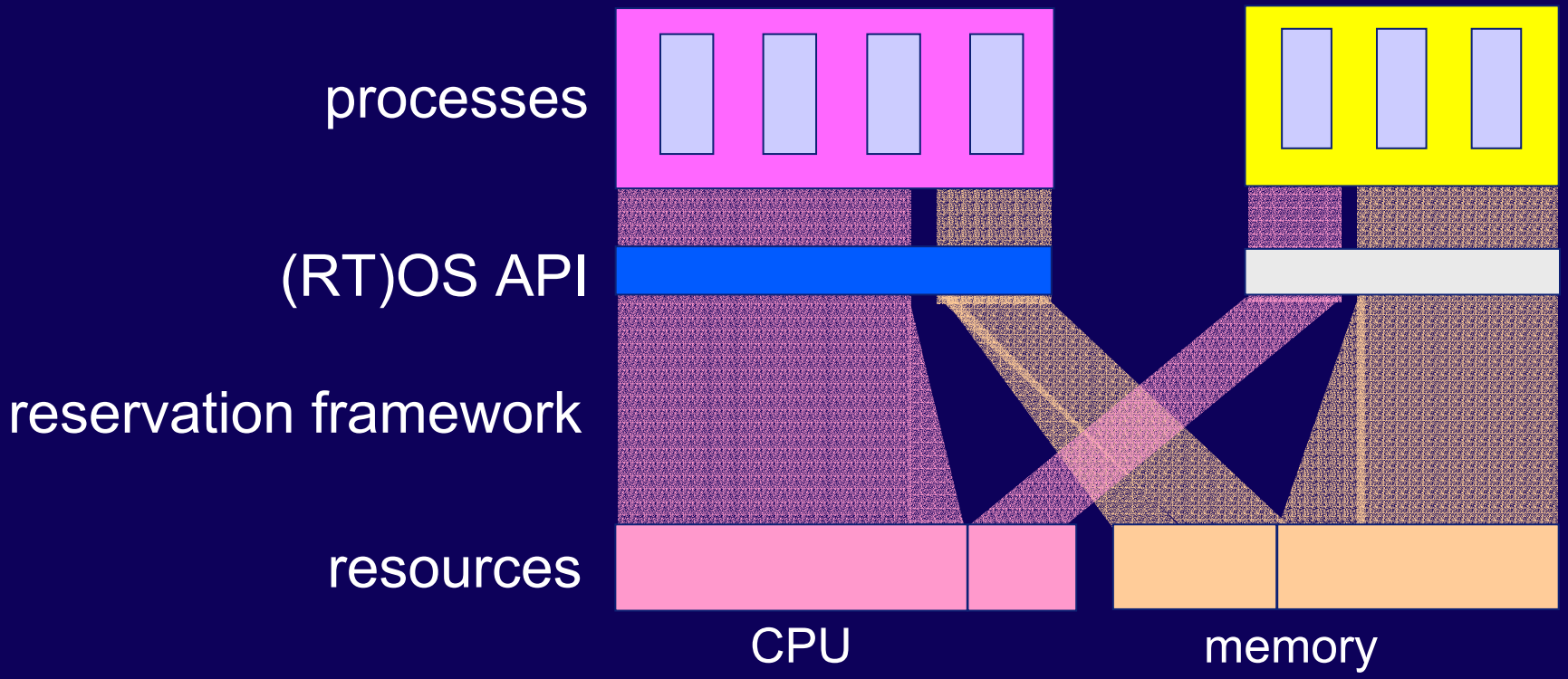
timesharing, lean, predictable



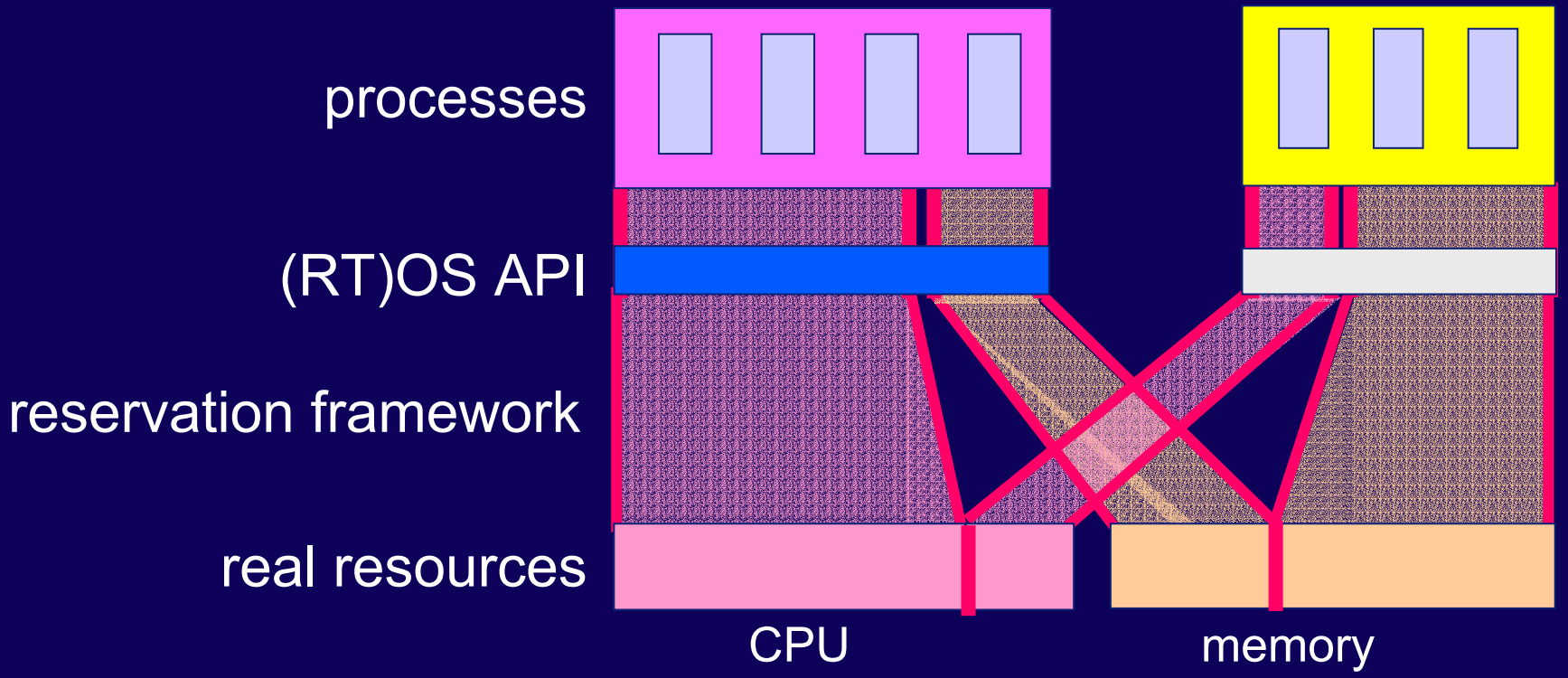
Approach 2

- **Proposition 1 Cluster of threads**
  - Provide reservations to clusters of threads rather than individual threads.
- **Proposition 5 RTOS API**
  - Make “classical” RTOS API available to cluster for local use.

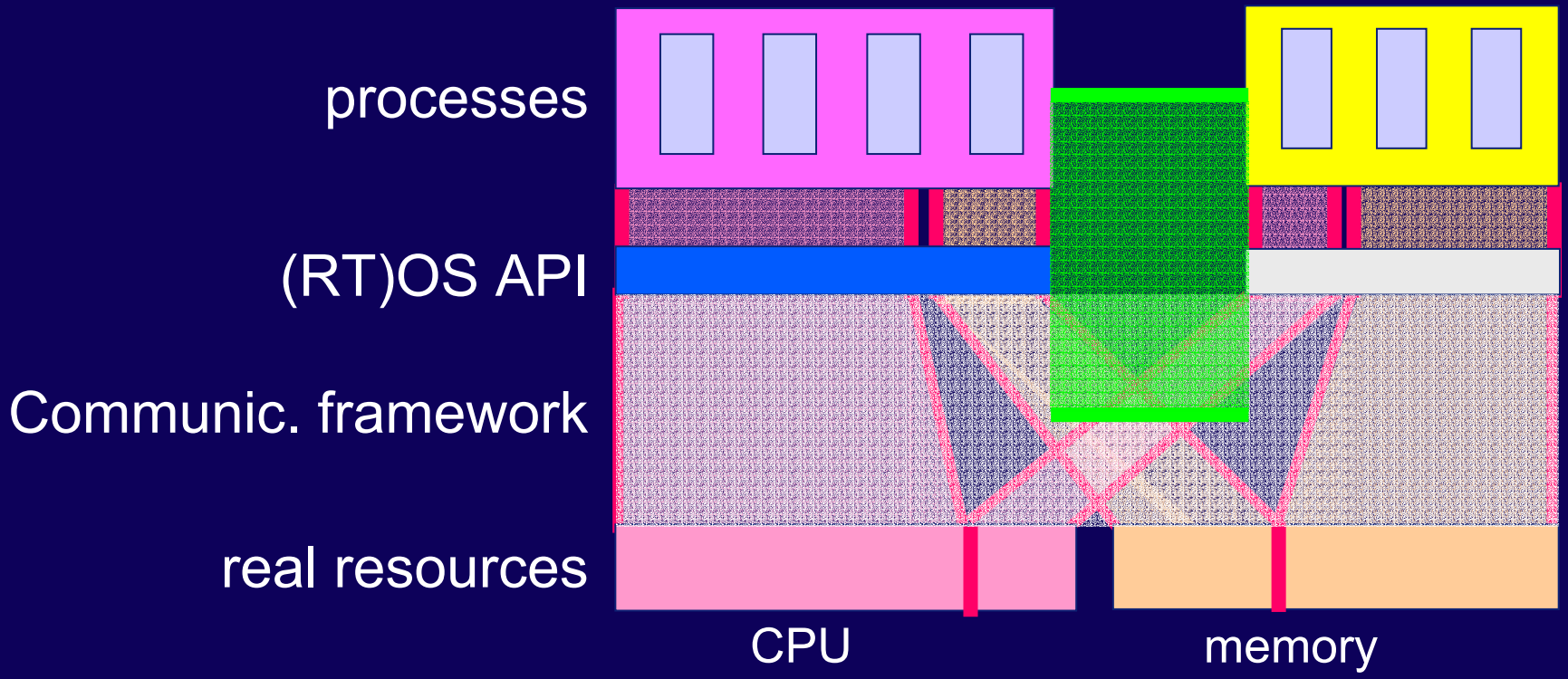
- **Proposition 2 Processor and memory**
  - Provide memory and processor reservations to same cluster.



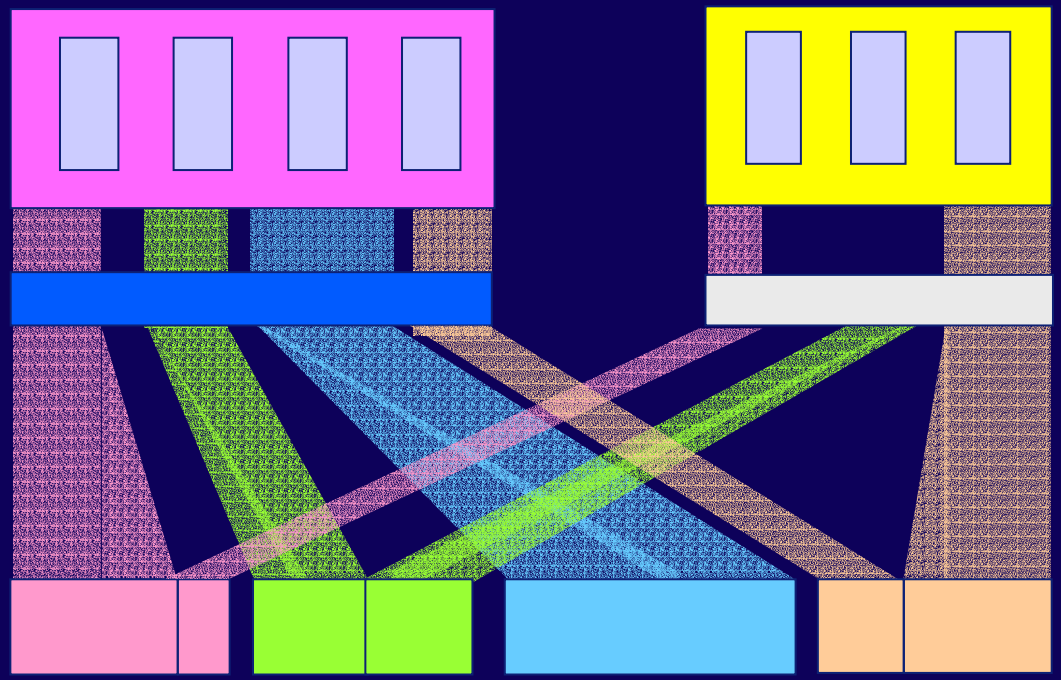
- **Proposition 3 Protection**
  - Make (temporal as well as spatial) protection an integral aspect of reservations.



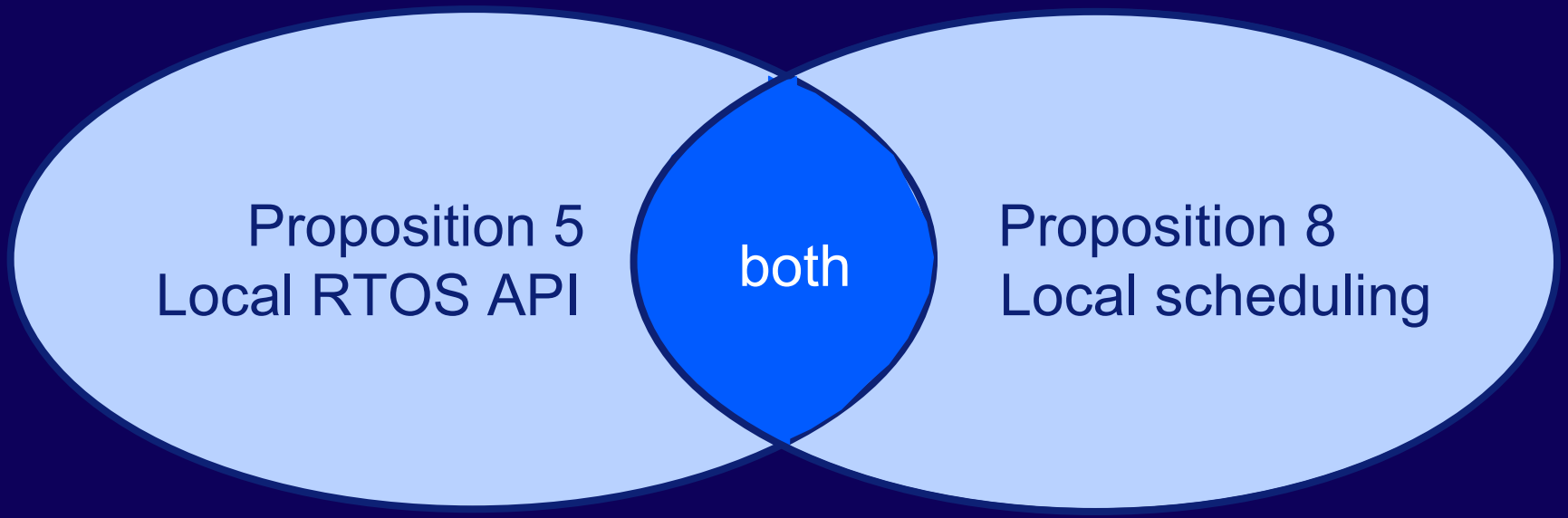
- **Proposition 4 Inter-cluster communication**
  - Provide primitives for inter-application communication with predictable temporal characteristics



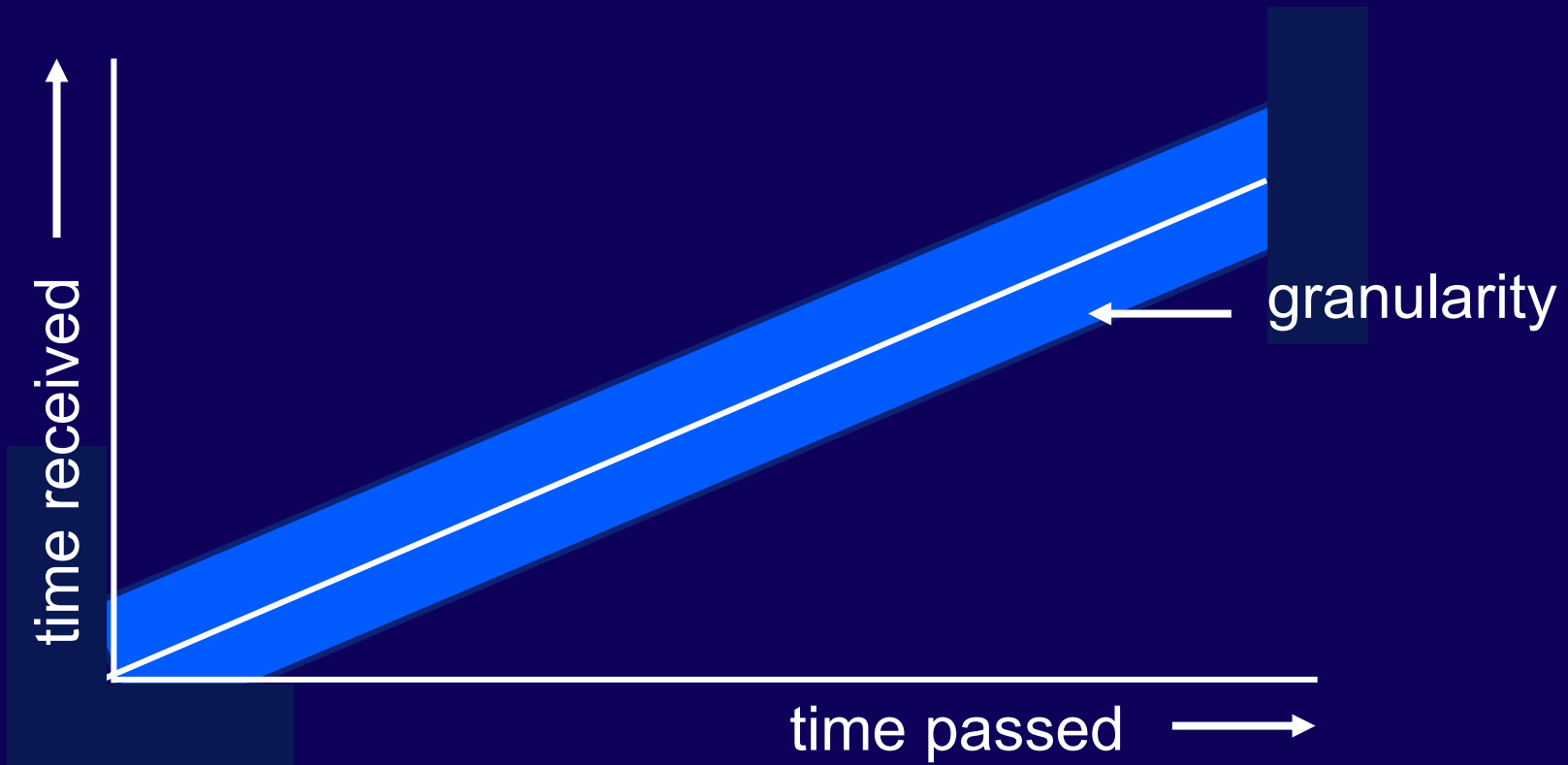
- **Proposition 16 Multiple resources**
  - Provide resource partitioning (in space and/or time) and associated protection for clusters as a unified strategy for all resources in a multi-resource environment.



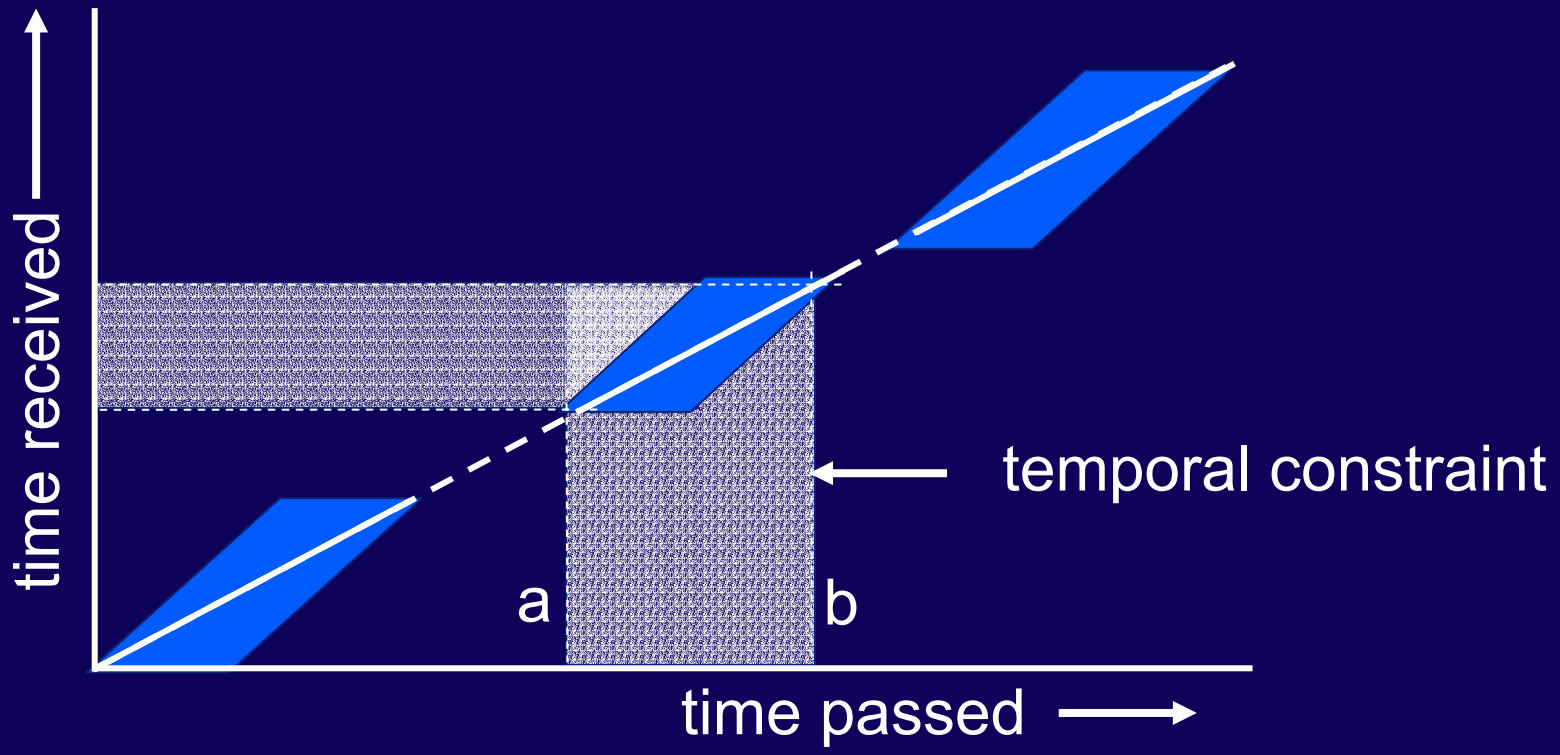
- **Proposition 8 Local scheduling**
  - Allow clusters to do their own local scheduling



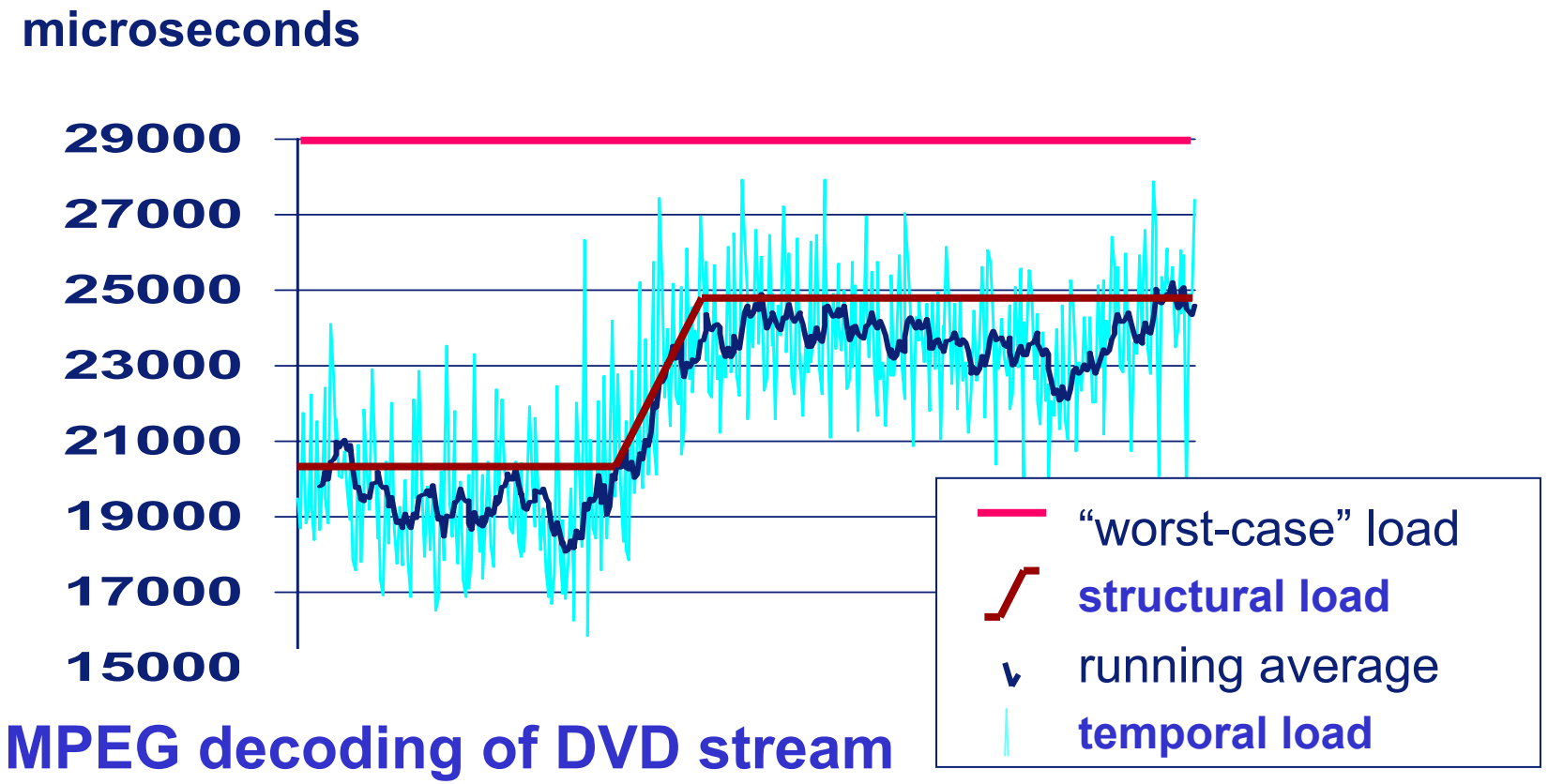
- **Proposition 6 Granularity**
  - Provide means for specifying allocation granularity in the reservation specification.



- **Proposition 7 Temporal Constraints**
  - Allow reservation contracts to specify **customised** temporal constraints, e.g. earliest start time (a) latest completion time (b).

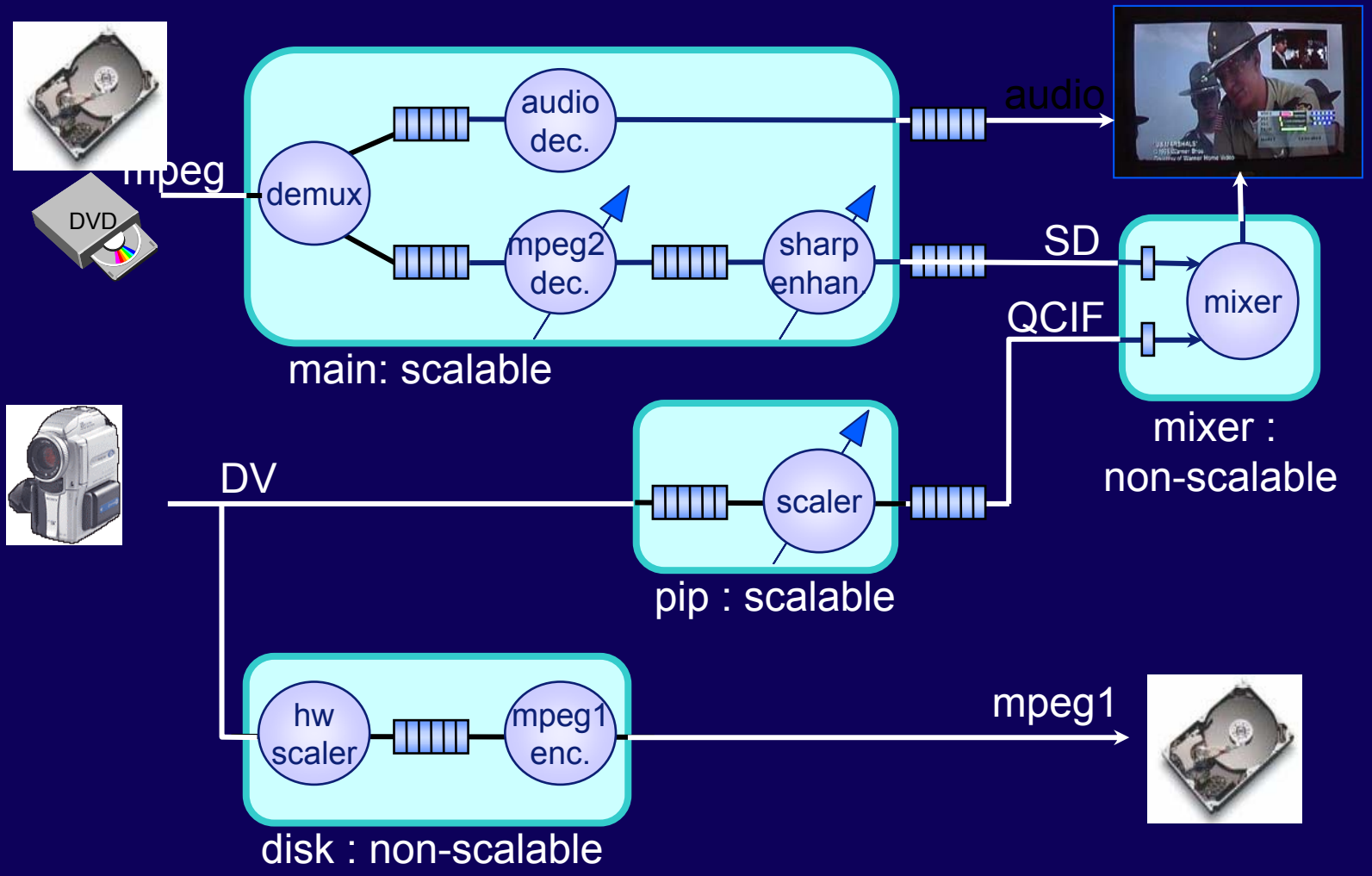


# QoS practice




**MPEG decoding of DVD stream**

# CE application for QoS experiments



# QoS practice - Dynamic control

- **Proposition 10 Resource monitoring**
  - The RTOS provides primitives for monitoring resource allocation and usage.
- **Proposition 13 Renegotiable reservation**
  - Allow renegotiation of reservation (or service) contracts.

# QoS practice – QoS-aware specifications

- **Proposition 11 Spare time**
  - Provide means to specify use of spare time in reservation contracts.
- **Proposition 12 QoS tolerance**
  - Allow reservation contracts for temporal resources to provide ranges instead of fixed parameters.

# Overhead

- **Proposition 18 Granularity Overhead**
  - Provide measures for weighing allocation granularity against the cost of context switching and cache flushing.
- **Proposition 20 System overhead**
  - Take system overhead into account when dimensioning a reservation (in the analysis phase).
- **Proposition 21 Interrupt handlers**
  - Account the cost of interrupt handling and RTOS services to the applications that effectively use them.

# Research recommendations

- **Proposition 9 Adaptive applications**
  - Investigate adaptive real-time applications.
- **Proposition 17 Multiple resources**
  - Investigate resource partitioning (in space and/or time) and associated protection as a unified strategy for all resources in a multi-resource environment.
- **Proposition 19 Cache issues**
  - Investigate cache issues in the context of sharing the memory access path.

- Our goal is to trigger a broad discussion between academic and industrial worlds on the steps that should be taken toward a new standard in real-time operating systems.

- We encourage scientist, practitioners and operating system's designers to join the discussion with a critical frame of mind, by reporting experiences, problems and suggestions.


PHILIPS